Statistical methods in the big data era
Hotel clustering

2019

EUSKAL ESTATISTIKA ERAKUNDEA
INSTITUTO VASCO DE ESTADÍSTICA
www.eustat.es

Organismo Autónomo del

EUSKO JAURLARITZA
GOBIERNO VASCO

Development:
EUSTAT
Euskal Estatistika Erakundea
Basque Statistics Institute

Publication:
EUSTAT
Euskal Estatistila Erakundea
Basque Statistics Institute
Donostia-San Sebastián 1,
01010 Vitoria-Gasteiz

Administration of the Basque Country

First Edition:
I/2019

Printing and Bounding:
Printing and Reprography Service. Basque Government.

ISBN: 978-84-7749-492-8

Legal Deposit: VI-157/19

Cover image: Marcos Gasparutti

STATISTICAL METHODOLOGIES IN THE ERA OF BIG DATA
Hotel Clustering

Asier Badiola Zabala

asier.badiola@outlook.es

EUSKAL ESTATISTIKA ERAKUNDEA
INSTITUTO VASCO DE ESTADÍSTICA

Donostia-San Sebastián, 1
01010 VITORIA-GASTEIZ
Tel.: 945 01 75 00
Fax.: 945 01 75 01
E-mail: eustat@eustat.eus
www.eustat.eus

Introduction

In recent times, “Big Data” has spread to all areas of society. It has also reached official statistics, where its use as a new data source represents a challenge. Various pilot projects are being carried out in statistics institutes in order to address the implications of “Big Data” in the different aspects of statistical production.

In 2016, Eustat organised the 29th International Statistics Seminar under the heading “Big data for Official Statistics”, given by Peter Struijs, coordinator of the Statistics Netherlands (SN) Big Data Program and coordinator of the ESSnet (European Statistical System network) Big Data group of the European Union.

This publication aims to publicise the research carried out in this field by one of Eustat training and research grants. This report consists of two parts. In the first part we will review the different machine learning methods applicable to Big Data, and in the second we will consider the classification study of hotel establishments in the Basque Country, which has been carried out on the basis of their price series scraped from websites.

Vitoria-Gasteiz, March 2019
Josu Iradi Arrieta
General Manager of EUSTAT

Contents

Preamble

This Technical Manual sets out the results of the work carried out on **machine learning**, thanks to the grant awarded in 2017 by the Basque Statistics Institute (Eustat) to provide training and conduct research into statistical and mathematical methodologies.

The main objective is to analyse and assimilate methodologies for processing information from different sources, without setting aside the data and statistics used to date. The aim is to offer more meaningful results.

We have used one hotel offer website to gain more information on tourism for this Technical Manual. Having attempted to contact the company and having received no reply, we initiated the *web scraping* process, once due notice had been given. As the data were being gathered (the process started in July 2017), they were crossed with data from Eustat's *Encuesta de establecimientos turísticos receptores*, identifying hotels and guesthouses.

Once the data had been combined, one of the main objectives of the grant was to classify the data in various ways, using different clustering procedures with the time series of prices. The results were presented on a poster at the *BigSurv18* conference, which was organised by the *European Survey Research Association* (ESRA) in Barcelona.

This manual is arranged into two main parts. The first section deals with the theory and different methods behind **machine learning**. The second section presents the work done on hotel clustering, while setting out the results.

I would like to take this opportunity to convey my appreciation to everyone at the Eustat Methodology, Innovation and R&D Department: Anjeles Iztueta, Jorge Aramendi, Elena Goni, Inmaculada Gil and Marina Ayestarán. My most sincere gratitude also goes to all of my colleagues at Eustat for creating an excellent working environment, and to my workmate Ander Juárez for putting up with me.

Gako-hitzak: machine learning, clustering.

Part I

Theory

Introduction

The first section will briefly and simply outline the various **machine learning** methods.The main idea behind machine learning is generally to create the most suitable methods based on the data gathered. However, we face several possibilities when creating these models. We can see many unknown parameters, to which we must assign certain specific values. Thus, machine learning seeks to *teach* the computer the most appropriate parameters in the most automatic way possible, hence its name.

Depending on the data to be processed, we can identify two main blocks: **supervised learning** (chapter 1) and **unsupervised learning** (chapter 2). In the first case, the data will carry a label, and the goal will be to create a model that labels data appropriately. In the second case, the data will have no label and the aim will be to determine their internal structure.

The chapter dedicated to **sequential data** (chapter 3), sets out a particular data structure in which those in a relevant order predominate. This type of data is often associated with time series. Typical examples include the stock market or the weather. The chapter will explain **Markov chains** and present a series of definitions and applications.

Artificial neural networks, which are currently gaining importance, also have their own chapter (chapter 4). These networks are used to create both supervised and unsupervised learning models, and have thousands of different applications. Although they were first developed several decades ago, their use has increased substantially in recent years as computer processing has accelerated.

The last chapter of this section (chapter 5) is dedicated to **ensemble methods**, which create a large number of simple methods that can be combined into a more solid model. While the concept might seem simple and straightforward, they are very useful for explaining complex structures efficiently.

Lastly, to find out more about machine learning, we recommend using the current R and/or Python programming languages. They are free pieces of software that offer an enormous variety of functions and packages. It is worth mentioning that the vast communities supporting them keep both programming languages constantly up-to-date.

Chapter 1

Supervised learning

The methods in this first chapter are divided into two main groups: regression and classification. The data must be *labelled* in both cases. In the first, the *label* must be continuous, whereas in the second it must be discrete and finite.

For example, we can observe the two tables below, where the data are identical and *labelled* (the *label* is shown in the last column). However, in the first case the label is continuous (price), whereas in the second it is discrete and finite (for rent/for sale).

m^2	Floor	Coastal location	Cost
110	4	Sí	400.000
80	1	No	190.000
150	2	No	330.000
...	...	...	...
70	5	Sí	390.000

Table 1.1

m^2	Floor	Coastal location	For rent/for sale
110	4	Yes	For rent
80	1	No	For sale
150	2	No	For sale
...	...	...	...
70	5	Yes	For rent

Table 1.2

In both cases, the aim is to develop a **model** capable of correctly *labelling* the new data being entered. Looking at the examples in the tables, in the first case we only need to appropriately estimate the price with different variables, whereas in the second we must try to predict whether the apartment is for sale or rent based on the input data.

Throughout this chapter, we will explain and analyse different methods for creating these models, as well as the methodology employed to test their quality in subchapter 1.3.

1.1 Regression

As we mentioned at the start of this chapter, the data show two types of variable: independent X variables on the one hand, and dependent Yvariables (referred to as a *label* in the preamble) on the other. The data corresponding to the independent variable may be both **quantitative** and **qualitative**, whereas the dependent variable will only be **quantitative** (in logistic regression, the dependent variable is qualitative, but logistic regression is regarded as a classification method). Depending on the type of variable, the method of obtaining the regression will be changed slightly, but the main aim will be to indicate the dependent Y variable through the independent X, i.e.:

$$Y \sim f(X, \boldsymbol{\beta})$$

For this we must find the function f and parameter $\boldsymbol{\beta}$in order to determine Y through X to the maximum extent possible. In practice, the definition of the function f is invariable, meaning that basic functions are used, such as linear functions, as we will see in greater detail in the next section.

The models will always produce an error ε, which we should generally seek to minimise. Once function f is defined, we will obtain the following statement:

$$Y = f(X, \boldsymbol{\beta}) + \varepsilon$$

and the work to be done will involve obtaining values for the parameter $\boldsymbol{\beta}$, so that the error ε will be as small as possible.

$$\text{Error} = \sum_{i=1}^{n} \varepsilon_i^2 \tag{1.1}$$

1.1.1 Linear regression

If the model is developed using linear regression, the dependent Y variable will be given as the linear combination of the n dependent variable $(X_1, \dots, X_n)$:

$$Y \sim \beta_0 + \sum_{i=1}^{n} X_i \beta_i \quad : \quad \beta_0, \beta_1, \dots, \beta_n \in \mathbb{R} \tag{1.2}$$

Taking up the example from the start of this chapter, we find three independent variables: $X_1 \equiv m^2$, $X_2 \equiv Floor$, and $X_3 \equiv Coastallocation.$ The parameters that need to be worked out to develop the model are therefore the values $\beta_0, \beta_1, \dots, \beta_n \in \mathbb{R}$.

These linear models are the simplest possible. There were, however, numerous methods developed before the computer age that are used to present the data simply and intuitively. Even nowadays, they carry considerable weight then developing probabilistic methods.

There are various ways and methods to solve the equation problem (1.2) (to obtain the values $\beta_0, \beta_1, \ldots, \beta_n \in \mathbb{R}$). In most cases, the independent X variables and dependent Y variables need to satisfy a series of conditions, including the error generated. We will not look at the statistics we can find following linear regression in more depth here. Nonetheless, there is an extensive catalogue of books (including numerous codes and packages in programming languages such as Ron statistics and linear regression. Books [10], [3] o [5] are the principal references.

When solving the problem (1.2) appropriately, we might find a result similar to the one given in figure 1.1.

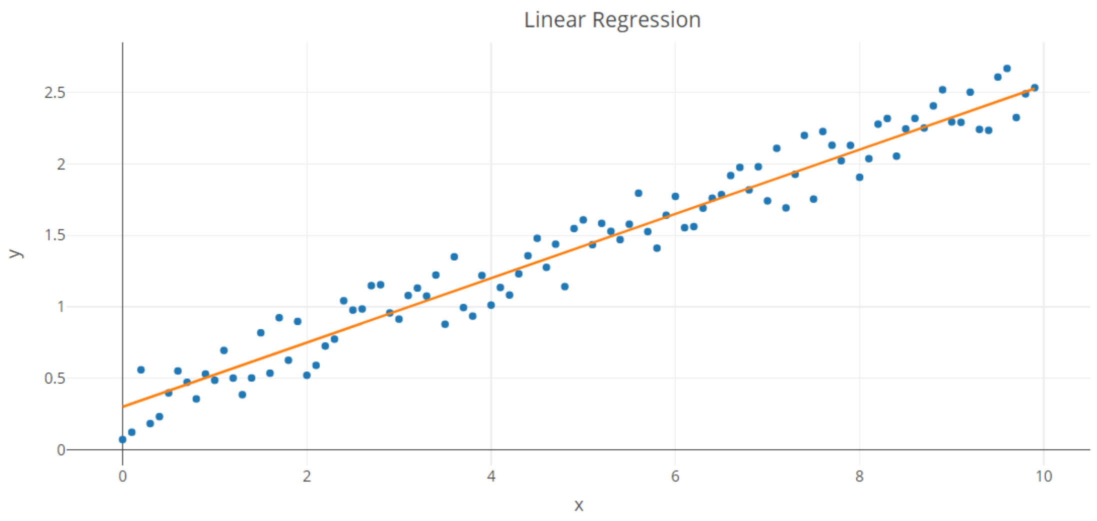


Figure 1.1: Example of linear regression. The independent x variable adopts values between $[0, 10]$, whereas the dependent y variable adopts values between $[0, 3]$. The model obtained is as follows: $y \sim 0.3 + 0.225x$. As we can see in the figure, there are different errors between the model generated and the data. The errors (ε) are often due to the *noise* produced by the data themselves and they may or not be acceptable depending on the case.

Error

Looking at figure 1.1, it is clear that the model obtained is not accurate and errors are thus generated. In this case, errors will always be produced because it is impossible to compile the dataset in a linear fashion. However, we can say at a glance that the model's definition of the dataset is reasonably satisfactory. The purpose of linear regression is to obtain a linear function that minimises this error, i.e. to select the values β_0 and β_i appropriate for the equation (1.2).

As we have said, the models that we developed with linear regression are very solid in theory, but to apply them, the data must satisfy a series of hypotheses, including the linear independence of the independent X variables, acceptance of the normality hypothesis by the errorr ε ($\varepsilon \sim N(0, \sigma^2 I)$), and so on.

More complex linear models

As we saw at the start of this subchapter with the model (1.2), we can obtain more complex and accurate approximations of the dependent Y variable by modifying the X variable and using the same method. Let us suppose that we have gathered the dataset from figure 1.2. As clearly shown in the first figure, $y \sim \beta_0 + \beta_1 x$ turns out not to be the most suitable model for identifying this group. However, if we use a third-grade polynomial, $y \sim \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$, the model will show the data trend with greater accuracy.

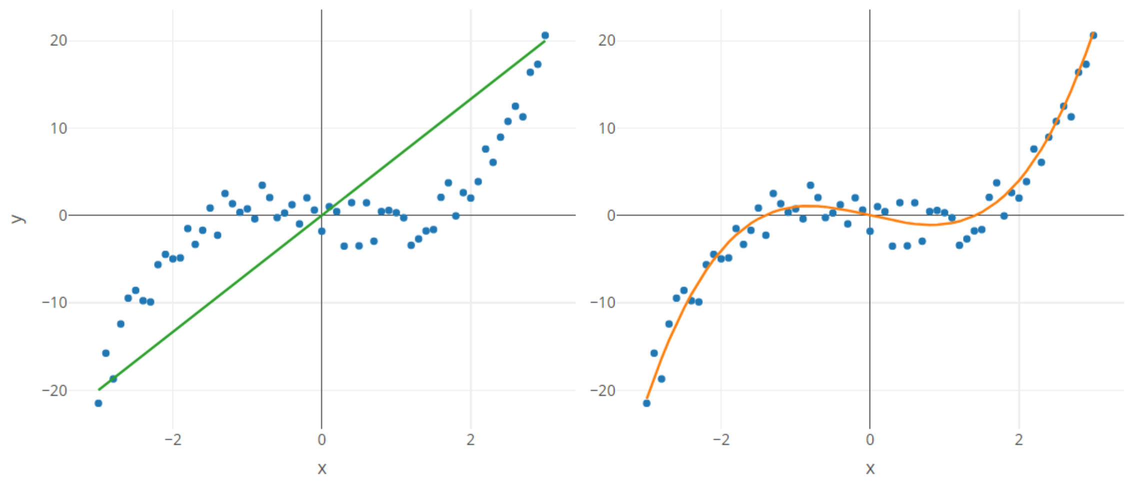


Figure 1.2: In the graph on the left, a simple linear model of the type $y \sim \beta_0 + \beta_1 x$ has been developed, whereas in the graph on the right we have a more complex model of the type $y \sim \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$. As we can see, the data trend is shown more accurately in the second model.

We might suppose that applying the exponent to the x variable might make it non-linear. This is not the case, however, because it is linear with respect to the variable $\boldsymbol{\beta}$.

$$Y \sim X \cdot \boldsymbol{\beta}$$

Although we continue to modify the X variable, the model with respect to $\boldsymbol{\beta}$ will still be linear. Thus, the most appropriate modifications ($\ln x, \sin x, \tan x \ldots$)) can be made to the independent X variable as long as they are linearly independent (see the example in table 1.3).

Regularisation

x	x^2	x^3	y
-2.5	6.25	-15.625	-21.5
-2	4	-8	-5
-1	1	-1	0.5
...	...	...	...
2.5	6.25	15.625	21

Table 1.3: In this case, the original input data are the grey columns, namely x and y. Another two variables are added to the table, modifying the x variable (through the exponents), so that the independent input variable $X = x$ gives the independent variables $X' = (x, x^2, x^3)$, with the intention of creating a more complex model.

Not all of the data are used for regression in the case of supervised machine learning. Rather, a significant portion is taken, while the remainder is used to measure the *quality* and *accuracy* of the model (see subchapter 1.3). This is why, when creating the model, we do not attempt to explain the data with much accuracy or detail, and instead examine the general structure, which can be done through **regularisation**. Various methods serve this purpose[1], but the most widely used involve establishing limits for the $\boldsymbol{\beta}$ parameters while penalising high values. As an example, the *Ridge Regression* model penalises these limits with Euclidean distance, meaning that the value to be minimised is:

$$\sum_{i=1}^{n} \varepsilon_i^2 + \lambda \sum_{i=0}^{n} \beta_i^2 \quad : \quad \lambda > 0 \tag{1.3}$$

To select the parameter λ, tests are conducted with different values. The values β_i are selected, which minimise the value of all those tried (1.3) and the results and model obtained are then tested. The model testing methods are explained in detail in subchapter 1.3.

Figure 1.3 shows the impact of regularisation.

The purpose of regularisation is to avoid excessively fitting the model to the data (we refer to such cases as **overfitting**). It penalises complex models and gives more weight to simplicity.

1.2 Classification methods

Once regression is complete, we will proceed with the classification methods. The input data will be in a space X, and each datum will be assigned to a single *class* or *label*. To show these classes, we will use the **discreto** o **cualitativo** space Y. Thus, if space Y is comprised of K classes, it will have the following form:

$$Y = \{1, 2, \dots K\}$$

[1]The methods can be analyzed in more detail in the section 3.4 de [15].

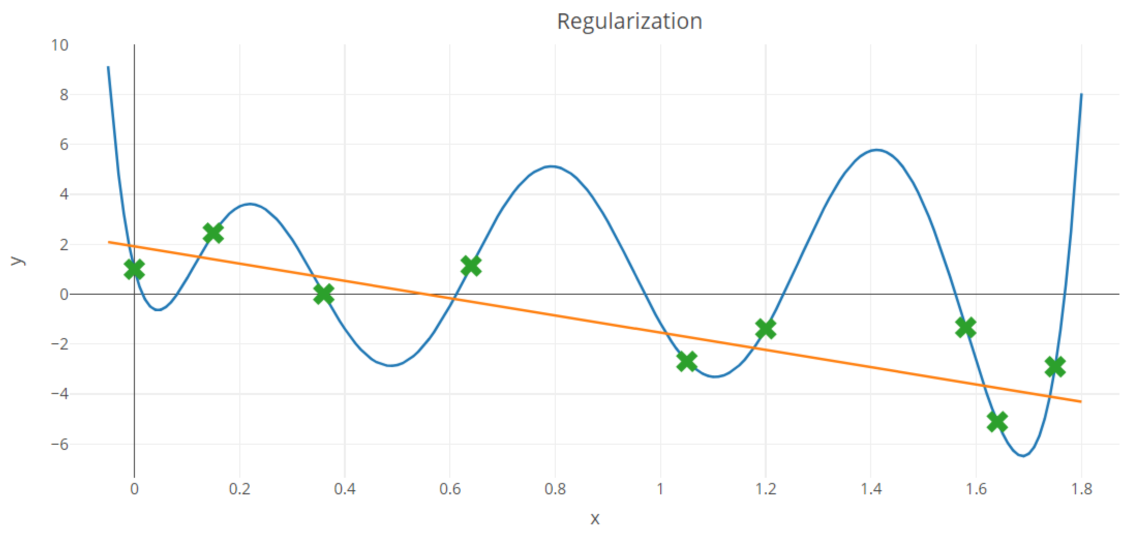


Figure 1.3: The green crosses indicate the 9 pieces of input data, whereas the blue line represents the model obtained using the eighth-degree polynomial. The value of the error contained in (1.1) will be 0, but it does not adequately explain the data. Using the true parameter λ, we obtain the simple linear model highlighted in orange, which would be more desirable.

The aim is to achieve a classification function f that assigns an appropriate class to any element in space X. If we analyse Figure 1.4, lthe input data are the points drawn. $X = \mathbb{R}^2$ is flat, and $Y = \{0, 1\}$ or $Y = \{urdina, laranja\}$, i.e. there are two distinct classes. Classifications with only two classes are called binary classifications, while those with more than two are called multiclass classifications.

The blue line in Figure 1.4 is the desired classification function f. We will obtain different classification functions depending on the method used and the results will therefore vary. Once we have found f, all of the points in space X will be assigned to a class. In Figure 1.4, the field highlighted in blue will be associated with one class, while the orange field will be attributed to the other.

1.2.1 k-nearest neighbours

The distance between elements in space X are used in this classification method. To classify space X, we assign each $x \in X$ the same class as the element it is nearest to, as we can see in Figure 1.6. In this case, the classification 1-NN is performed, meaning that only the class of the point closest to it is considered. It is worth pointing out that the least distance must always be calculated with respect to the input data.

However, this case can be generalised. Instead of considering only the element closest to element x, k nearest elements can be considered. Element x will be assigned to the class most frequently occurs among the elements, as will be explained below. This is called k-NN classification.

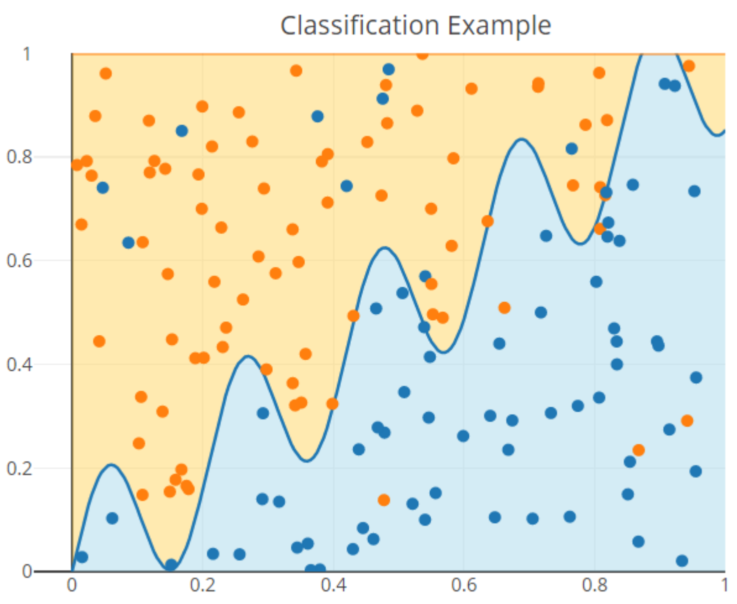


Figure 1.4: Example of a classification method. The orange and blue points are the input data, each with their corresponding label or class. The blue line is the classification model created. Points above the line will be attributed to one class, while those below will belong to the other

Selecting the k value

With the k-NN classification method, the error will generally be small when the value of k is low. When the value of k is higher, the classification areas tend to be *smoother* and more *estable*, as Figure 1.7 shows. To select the appropriate value of k, the techniques from subchapter 1.3 are used to develop various models for different values of k and then test the results. Selecting the k values can be viewed in the same way as regularisation in the previous section: lower k values may result in **overfitting**.

Advantages and disadvantages

As we have said, the method is simple and easy to understand and implement; one of the few elements to be defined is the distance that we want to use. There is one drawback, however. If the input dataset is large, the algorithm will have a high cost since, in the k-NN method, the distances between all the points have to be calculated.

1.2.2 Decision trees

The (*decision trees*) method can be used for both **classification** and **regression**. In this section we will focus on classification. This method is based on the *division system* and, to make it as intuitive as possible, we will start with an example.

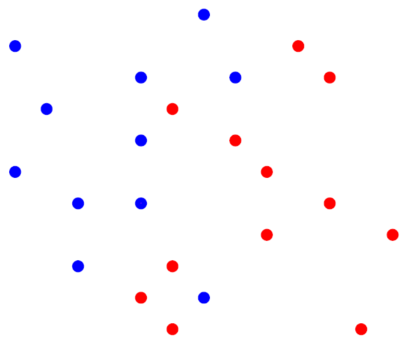


Figure 1.5: Binary classification of input data.

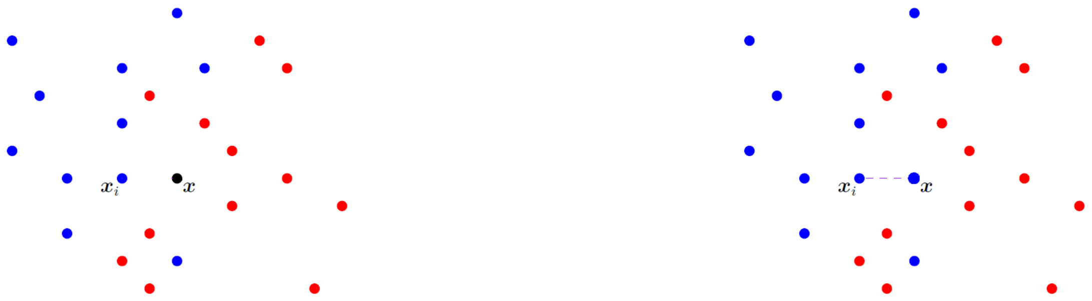


Figure 1.6: The class of element x will be the same as the element nearest to it (in the case of 1-NN). On this occasion, therefore, x will be classified in the blue class.

Figure 1.8 displays a 4 classes dataset where there are two dimensions, X_1 and X_2.The tree on the right shows the route followed for classification. Firstly, point t_1 is obtained on the X_1 axis, placing the points to its right in group A (yellow points). Point t_2 is then obtained on the X_2 axis, placing the points below this value in group B (red points). Lastly, t_3 is obtained on the X_1axis, creating groups C and D (green and blue points, respectively).

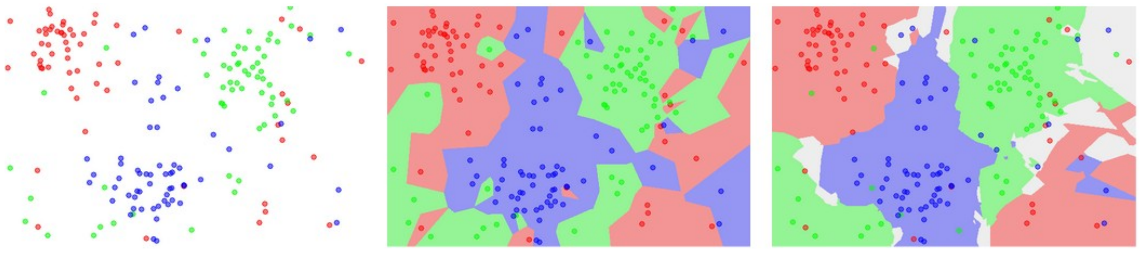


Figure 1.7: Original data (a total of 3 classes) appear in the figure on the left; the middle figure gives the 1-NN classification; and the figure on the right shows the 5-NN result. The grey spaces on the right represent the two most frequent classes.

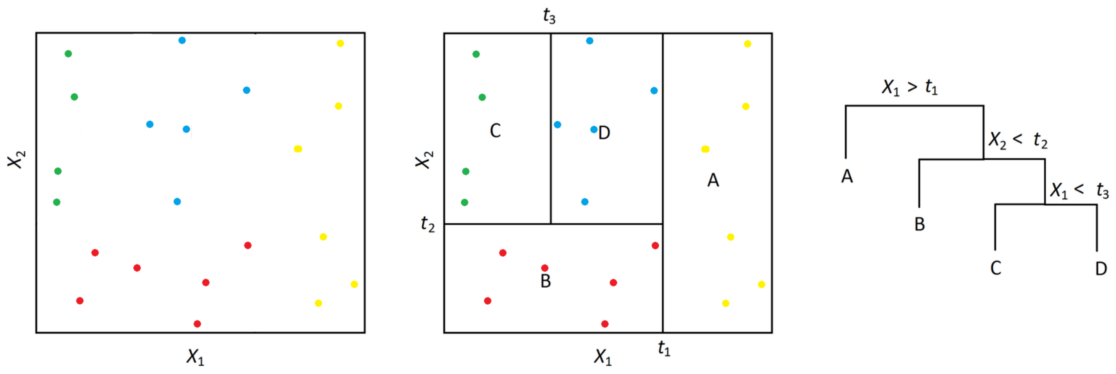


Figure 1.8: The figure on the left shows 21 pieces of input data classified into 4 classes. The figure on the rights displays the classification obtained after applying the decision tree method.

Advantages [2]

- Straightforward and simple to understand and interpret.
- It can handle both quantitative and qualitative variables, and does not require extensive data pre-processing.
- The method can handle large quantities of data.

Disadvantages

- Decision tree algorithms can be unstable, and minor changes in the input data can give very uneven results.
- Achieving a globally optimal decision tree is an *NP-complete*[3] problem [11]. Ensemble methods are used to tackle this computational issue. (For further information, see chapter 5.)

[2]For more detail about the advantages and disadvantages: [27] `http://scikit-learn.org/stable/modules/tree.html`

[3]To understand the general idea about a problem *NP-complete*: `http://www.geeksforgeeks.org/np-completeness-set-1/`

- The method's simplicity entails a series of problems when explaining more complex concepts: the XOR/ALA logic threshold, multiplexers, etc.

The idea behind the algorithm

The idea behind the algorithm is to find the point that reduces the classification error in the variables. For example, in Figure 1.8, after various tests on the variables X_1 and X_2 using the algorithm, it is concluded that the point that minimises the classification error is t_1 of the variable X_1. To measure the error, the **entropy** or **Gini index** is normally used. Once the point t_1 has been defined, the same procedure is followed to obtain points t_2 and t_3.

However, this algorithm can sometimes fail, depending on the disaggregation of the input data, as reflected in Figure 1.9. A common way of solving this problem is the **Pruning** method, which involves obtaining a large initial tree to then eliminate or prune its *nodos* and *branches*, until a smaller tree is achieved that better suits our needs. This method is also used as regularisation to avoid overfitting.

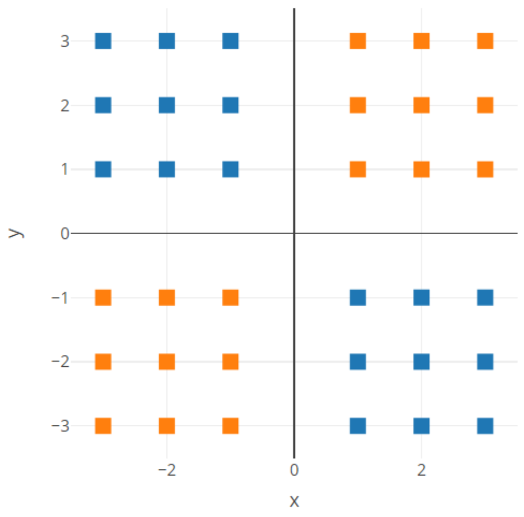


Figure 1.9: In this graph, constructing the tree might appear simple at first glance. However, the algorithm would be unable to select the appropriate points t_1 and t_2 because it cannot select the point that minimises the error. The **Pruning** method is used in such cases.

1.2.3 Support vector machine (SVM)

To finish with the classification methods, we will look at the **Support Vector Machine(SVM)**. The SVM method was developed in the field of computing in the 1990s and has gained in popularity over the years. It has been shown to yield good results with different groups of data and is widely considered one of the best simple-to-use methods.

To understand its functionality, we will first give the intuitive definition of the hyperplane and, with this, we will analyse the main ideas behind the algorithm known as **maximum margin classifiers**. Once this point is defined, we will explain **support vector classifier** and, finally, the **SVM** method.

Maximum margin classifiers

To use this algorithm or technique, we will suppose that the data to be analysed have a binary classification and are **linearly divisible**[4]. The fact that they are linearly divisible implies that, mathematically, the points created can be divided by a hyperplane. A two-dimensional space can be divided with a straight line, whereas a three-dimensional space can be divided by a plane.

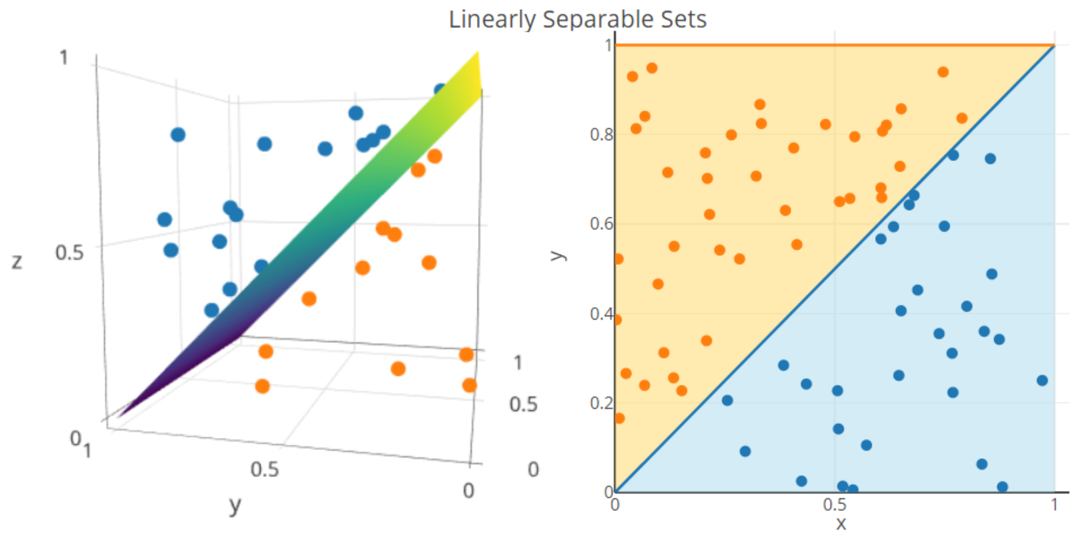


Figure 1.10: Both groups of data in the figure are linearly divisible. In the first, a separating plane in a three-dimensional space has been created, while the two-dimensional space has a separating line.

With linearly divisible groups of data, there will in most cases be infinite hyperplanes to carry out the classification. Let us take Figure 1.11 as an example. Both lines are valid for classification but, in this case, the second classification will have the most weight and, in some ways, it will be *better*.

Maximum margin classifiers seek the second result, namely the hyperplane that is the **furthest** from each class. The intuitive concept used to maximise the distance is shown in Figure 1.12.

Firstly, a convex group of points from each class is created and the hyperplane furthest from them is found, i.e. the hyperplane at the centre of those groups.

The hyperplane is achieved mathematically as a solution to an optimisation problem, as set out below:

[4]In chapter 9 of the book [17] you can read the details about the SVM method

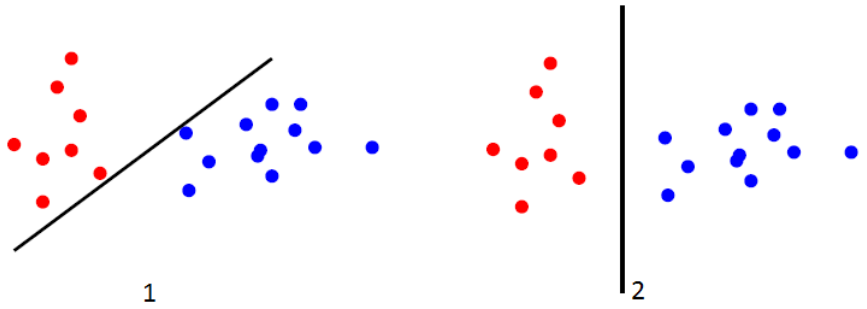


Figure 1.11: Two different classifications of the same linearly divisible dataset.

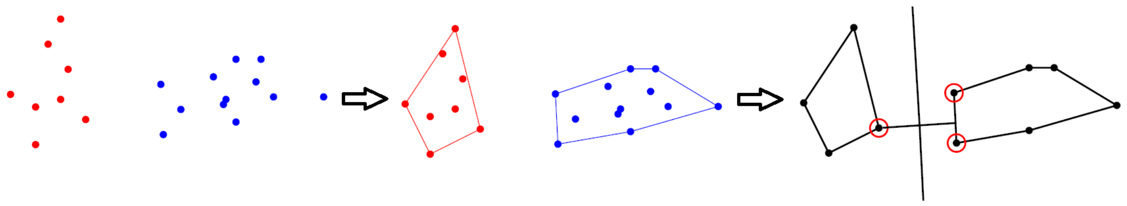


Figure 1.12: Intuitive concept behind the maximum margin classifier.

- Where the input data are $x_1, \ldots, x_n \in \mathbb{R}^p$ and the binary class that corresponds to those data is $y_1, \ldots, y_n \in \{-1, 1\}$.
- The goal is to obtain a hyperplane $\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p = 0$ that satisfies the conditions above. For this we have to select the appropriate β_i values for the values of $i = 0, 1, \ldots, n$, which are obtained by solving the following optimisation problem:

$$\max_{\beta_0,\beta_1,\ldots,\beta_n} M$$

associated with the two conditions

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i_1} + \ldots + \beta_p x_{i_p}) \geq M, \quad \forall i$$

Support vector classifier

The idea behind the support vector classifier is identical to the maximum margin classifier, which is to select a *suitable* hyperplane for classification. However, in the latter case, the input data do not necessarily have to be linearly divisible, meaning that it can be applied in more general cases. For this it has to be assumed that the classification may produce errors. The base method will still be the same nonetheless.

In this case, the problem to be optimised is:

$$\max_{\beta_0,\beta_1,\ldots,\beta_n,\epsilon_1,\ldots,\epsilon_n} M$$

associated with the conditions

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i_1} + \ldots + \beta_p x_{i_p}) \geq M(1 - \epsilon_i), \quad \forall i$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$$

As we can see, the main difference lies in the appearance of the values ϵ_i. This permits classification errors, in which case the C value will limit them (we can regard the C value as a regularisation parameter).

Support vector machine

Classifying data with a hyperplane will often suffice. However, we will not generally be able to divide the data gathered in a linear manner because their classification is more complex, as can be seen in Figure 1.13. These are cases in which we use SVM, an extension of the support vector classifier for which *kernel* come into play.

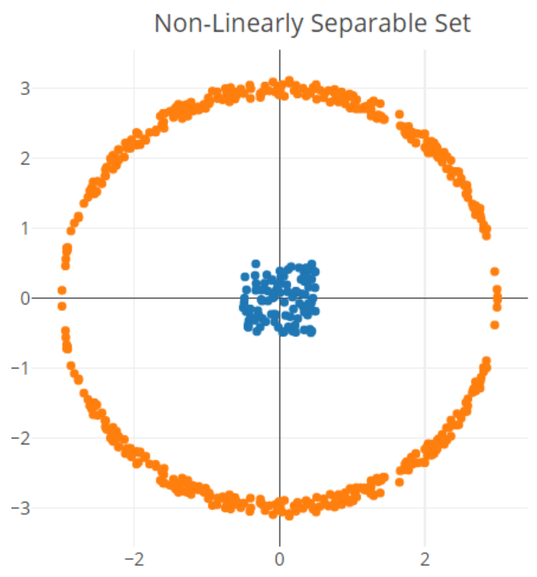


Figure 1.13: Group of data that cannot be divided linearly.

It is worth considering that in this next part we will discuss the underlying **concept**behind the SVM method, owing to the complexity of its mathematics and theory, and because the purpose of this manual is not so much to look at the technical aspects in depth. As we have said, the use of different *kernel* forms the basis of SVM and we recommend that those who wish to further explore the topic should start with section nine of the book [17], as it provides a simpler explanation for understanding this section.

We will continue with the example from Figure 1.13. We cannot find a suitable hyperplane to enable us to make an adequate classification. We should recall that in this case the hyperplane will be a straight line. Thus, the data will change space in order to come up with a solution to these cases. For this we will define a function ϕ, where $\phi : \mathbb{R}^p \longrightarrow \mathbb{R}^P$ will take the data to a higher dimension. We can analyse graph 1.14 on the left as an example. The data $\mathbb{R}^2$ in Figure 1.13 are transferred to space $\mathbb{R}^3$, through the following function ϕ:

$$\begin{aligned} \phi : \ \mathbb{R}^2 \quad & \longrightarrow \quad \mathbb{R}^3 \\ (x, y) \quad & \longrightarrow \quad \phi(x, y) = (x, y, \sqrt{x^2 + y^2}) \end{aligned}$$

Once the data are transferred to this new space, we have the option of creating a hyperplane capable of making a suitable classification. In this example, let us consider the hyperplane $z = 2$. Once the hyperplane is obtained, we return to the original space to analyse the classification field achieved (graph to the right of Figure 1.14).

$$z = 2 \Longrightarrow 2^2 = x^2 + y^2$$

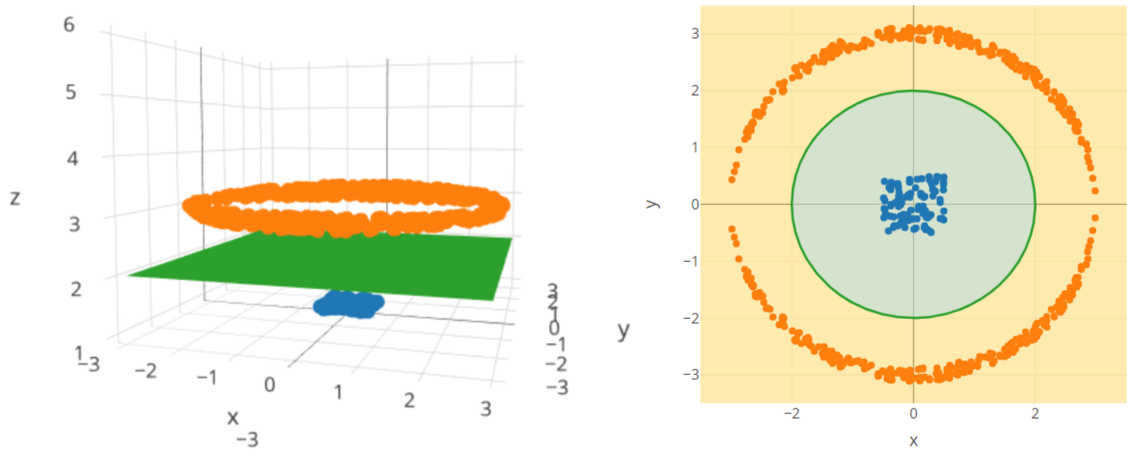


Figure 1.14: In the figure on the left, the original data go from the second dimension to the third via a function ϕ, and are then classified using a hyperplane. The figure on the right shows the final classification obtained through separation by the hyperplane.

It is worth remembering that this explanation of SVM is intended to be intuitive and simple. We should take into account, first and foremost, that the different *kernel* used in SVM mean that the algorithms and techniques that we have seen both with the support vector classifier and the maximum margin classifier are suited to cases that are not linearly divisible. Using the appropriate *kernel* allows us to make as complex classifications as we wish.

1.3 Testing the models

Thus far, this chapter has looked at the various supervised machine learning methods for developing models. Nonetheless, we have seen that there are several different ways of creating these models and that the classification or regression methods thus obtained may differ substantially. With this in mind, we need to have a system of choosing the most appropriate or suitable method from all the different options.

When developing the models, we must bear in mind that in most cases, they must be capable of correctly classifying future data. We can take Figure 1.15 as an example. It shows two graphs. The green crosses represent the input data used to develop the models, while the orange and blue lines are two distinct models. If we look at the graph on the left, the blue line distributes the data without any error, whereas the orange line presents a series of errors. When entering new data as shown by the red crosses in the graph on the right, we notice that the orange model gives a more appropriate indication of the data's behaviour, meaning that the error is lower.

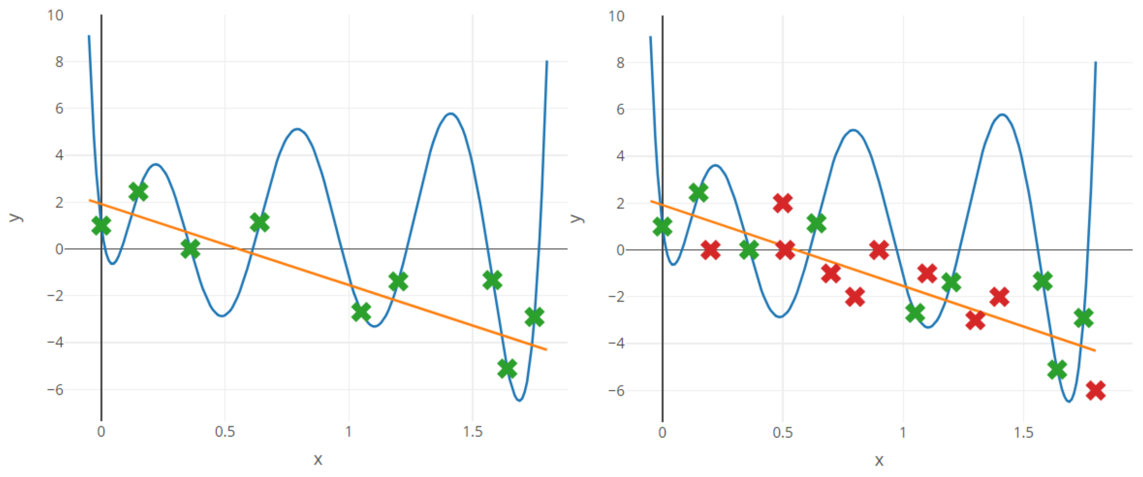


Figure 1.15: In the figure on the left, we can see two different models (orange and blue) developed using the input data (green crosses). New data are shown in the figure on the right (red crosses).

However, because in many cases we will not be able to wait for new data to test the models, the data available are used to conduct the tests. This is a relatively simple idea that involves using some of the data for training and some for testing. In other words, some of the data will be used to develop the model, while the rest will be regarded as *new data* with which to conduct tests.

It is very important for the division to be as uniform as possible in order to collect the maximum data variation. For example, if we want to develop a model with the different eating times in Europe, it would make no sense to leave out data on Spain when developing the model and later use them to test it.

1.3.1 Training, validation and testing

To test the models, the data are normally divided into three parts, namely the *training set*, *validation set* and *test set*. Some general guidelines are followed when dividing the data. If there are around 100, 1,000 or 100,000 pieces of data, 70% or 80% of them are included in the *training set*, while the remaining 30% or 20% are distributed among the *validation* and *test sets*. For cases where the quantity of data is higher – more than 1,000,000 elements, for example – approximately 98% is used for the *training set*, 1% is used for the *validation set* and the remaining 1% is used for the *test set*.

Training
Validation
Test

The models for both regression and classification are developed based on the *training set*. In some cases, however, the models fit the input data too closely (overfitting), meaning that the results will not be valid for new data. A clear example of this is the blue model in Figure 1.15.

We use the *validation set* precisely to avoid such cases. Different models are tested using this dataset to determine whether or not there is overfitting. To select the model, a table or graph is designed with the errors from the *training set* and the *validation set* (Figure 1.16), attempting to minimise the errors as much as possible.

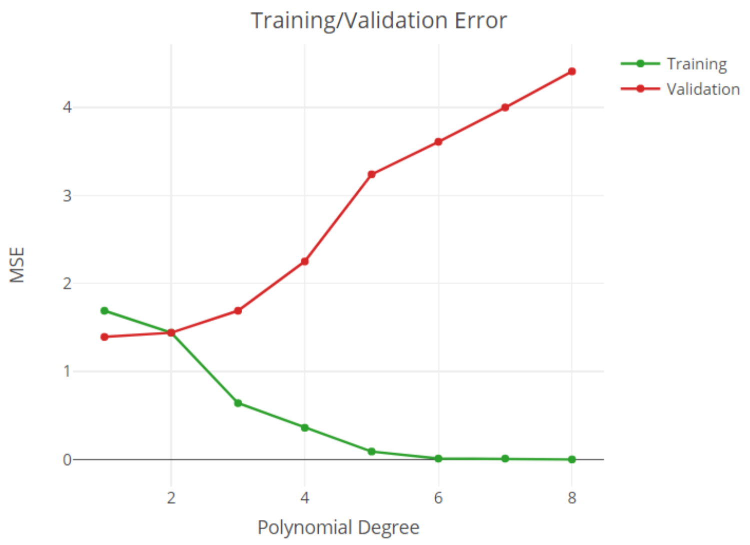


Figure 1.16: This graph contains the errors obtained from the data in Figure 1.15 (mean squared error), depending on the polynomial function employed. As we can see in Figure 1.15, the first-grade polynomial gives a larger error with respect to the eighth-grade polynomial in the *training set*, but a smaller one in the *validation set*.

As Figure 1.16 shows, as the model becomes more complex, the error produced by the *training set* decreases, owing to its ability to place the data directly. On the other hand, if we analyse the error given by the *validation set*, we will see that it increases considerably from the third level. This is overfitting expressed on a graph.

Because, as we said, the goal is to avoid overfitting and minimise errors, we will select the model that contains one or two levels. The model that minimises the error is generally selected from the *validation set*. If there are several models that satisfy this condition, we will select the one that gives the smallest error in the *training set*.

Once selected, we will use the *test set* to measure its quality. Unlike the *validation set*, it is only used to calculate the final error and not to select the model.

For more information about tests, see chapter five of book [17], which provides details on tests such as **cross-validation**, and on various ways of measuring the error.

Chapter 2

Unsupervised learning

Unsupervised learning is a machine learning task that infers the hidden structure of *unlabelled* data. The data lack any type of class, category or classification and the evaluation methods of the models constructed from the data are more subjective compared to supervised learning.

There are algorithms and methods of different kinds, but there are generally two main tasks to perform on *unlabelled* data: **grouping them** and **reducing their dimensionality**.

The most well-known in the first case is **clustering**, as we will see in subchapter 2.1, where the data are grouped by their common characteristics. Another well-known case is the **detection of anomalies**. The purpose of these methods is to identify the datum or dataset that are furthest from the compiled data or that seem anomalous. Another two subchapters will be included in this manual on the grouping of data: **matrix factorisation** (subchapter 2.3) and **association analysis** (2.4).

One of the examples of **data dimensionality reduction** is provided in the **principal component analysis** (subchapter 2.2). The aim of data dimensionality reduction is to broaden their interpretation and reject unnecessary information. To put it another way, intelligible and simple data are prioritised.

As we will see in this section, unsupervised learning models can be looked at in the same way as supervised learning models, with some minor changes. That said, other techniques can also be used to measure the quality and fit of the models.

2.1 Clustering

When forming clusters, the data are provided without any type of *label* or classification, and the aim is to group them according to their *proximity* or *similarity.*

Expressed mathematically, the data $x_1, \ldots, x_n$ belonging to the space X should be classified into $K \in \mathbb{N}$ different groups, entering nearby points into the same cluster. To make things easier, let us suppose that the variables are continuous and that we have a vector space $X = \mathbb{R}^d$ (in the examples, we will use $X = \mathbb{R}^2$).

Selecting the number of groups that we want to create can be quite subjective and should be considered on a case-by-case basis. Sometimes, however, the number of clusters is predefined (let us suppose, for example, that we are asked to assign a score from 1 to 5 to the level of vegetation in each municipality using measurements taken in the Basque Country). A way of selecting a number of clusters K is explained in the method below.

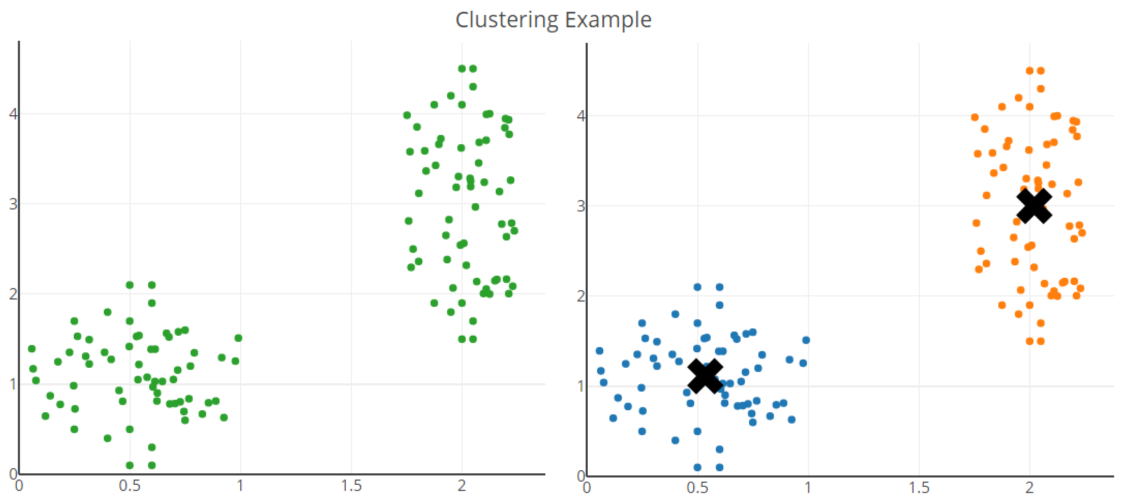


Figure 2.1: The figure on the left shows a set of input data, while the data are divided into two clusters in the figure on the right. The two black crosses appearing in the figure on the right are centroids μ_1 and μ_2 of the clusters mentioned.

$\boldsymbol{K}$-Means method

One of the most straightforward, simple and widely used methods for forming clusters is **$\boldsymbol{K}$-Means**. Its principal idea involves adding the points around certain points μ_k (called centroids) to the group k. The centroids are defined as follows:

$$\mu_k = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i \mathbb{1}_{\{c_i = k\}} \in \mathbb{R}^d \tag{2.1}$$

where n_k indicates the number of elements in group k, $\mathbb{1}$ is the function indicator, while $c \in \{1, \ldots, K\}$ constitutes the indicator of the group:

$$c_i = k \Leftrightarrow x_i \text{ belongs to the cluster } k \tag{2.2}$$

The goal of the algorithm is to find an optimal $\boldsymbol{c}$ $=(c_1, \ldots, c_n)$ and $\boldsymbol{\mu}$ $=(\mu_1, \ldots, \mu_K)$, so that all of the points stay as *close* as possible to the cluster. In mathematical terms, we are trying to determine:

$$\underset{\boldsymbol{\mu}, \boldsymbol{c}}{\arg\min} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}_{\{c_i = k\}} d(x_i, \mu_i) \tag{2.3}$$

where d measures the distance to be used.

However, this minimisation problem is *NP-Hard*[1] [11], i.e. of high computational complexity. To address the problem, other simpler algorithms are used that achieve **local minimums**. One of the most widely used is **Lloyd's algorithm**, whereby centroids are obtained through iteration and two basic steps are followed: redefining the centroids and groups (until convergence).

Hundreds of videos and animations can be found on the internet to help give us an intuitive understanding of the algorithm. This link is included as an example:

`https://www.youtube.com/watch?v=5I3Ei69I40s`

However, it should be borne in mind that, in general, the value for (2.3) will decrease as the value of K increases. One of the ways of selecting K is thus to increase its values individually and, when the value of (2.3) increases by a *small* amount, we will be left with the last value of K. If we look at Figure 2.2, we will see that in this case it is advisable to create three clusters, since any improved error from that point is considered minimal. However, this method should be used as a guide, as more or fewer clusters can be created depending on the objectives pursued.

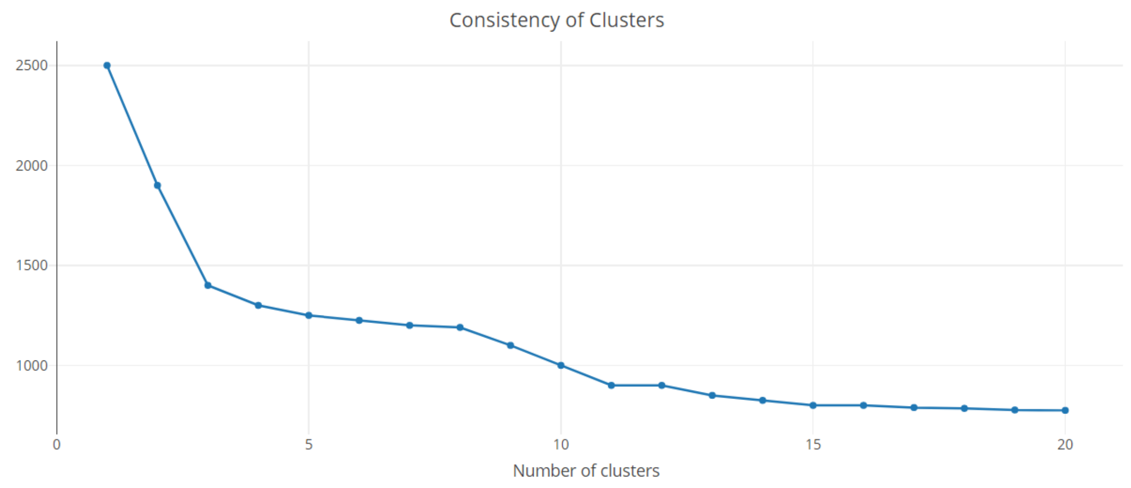


Figure 2.2: Graph analyses density, i.e. the closeness of the elements to each cluster, as the number of clusters K is increased.

Probabilistic methods

There are also **probabilistic methods** for creating clusters. The clusters it seeks to create are identical, i.e. obtaining K groups based on data, but it makes a series of assumptions that lead on to the use of statistical inference.

If we assume that data come from different normal distributions – this is known as the **Gaussian Mixture Model** (GMM)[2] – the breakdown of the clusters that

[1]The general idea about what is a problem *NP-Hard*:
`http://www.geeksforgeeks.org/np-completeness-set-1/`

[2]You can consult section 9.2 of the book [2] for more details.

we want to obtain will be of the type $N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$.

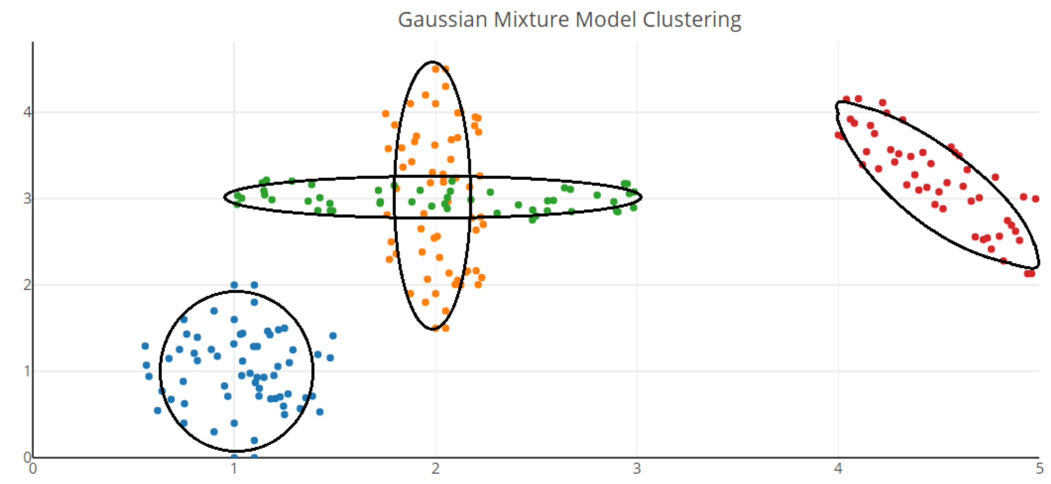


Figure 2.3: In this graph, we can see that there are 4 clusters and that it assumes that the points for each give a normal distribution $N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$.

The positive aspect of GMM is that statistical inference can be used to obtain more solid structures and conclusions. However, the clusters will have a certain shape. Looking at Figure 2.3, the clusters obtained in two dimensions will be ellipsoid because they are based on normal distribution.[3]

2.2 Principal component analysis

Principal Component Analysis (PCA) is a method used particularly to reduce the dimensionality of the input data. Let us suppose that the input data are $x_1, \dots, x_n \in \mathbb{R}^d$, while d represents the number of variables in each element. On many occasions, some of the dimensions or variables offer little new information, or there is some collinearity among them. It is therefore desirable to eliminate the unnecessary variables to save memory and reduce the computing time.

Figure 2.4 gives an intuitive depiction of the PCA algorithm in action. The input data are four-dimensional: three are spatial and one relates to colour. The aim on this occasion is to classify these points, for which we have two options. On the one hand, we have the option of starting work with the desired classification method directly. On the other, we can apply dimensionality reduction followed by the classification method. As we can see in the figure on the right, a spatial dimension – a considerable amount of information – has been eliminated and yet the classification can be carried out adequately.

The PCA is based on the decomposition of eigenvalues[4]. In the case of developing

[3]The Python programming language offers several modules to form clusters using different methods [27]: `http://scikit-learn.org/stable/modules/clustering.html`

[4]You can consult about the decomposition of eigenvalues in most linear algebra books, as for example in section 5 of [33]

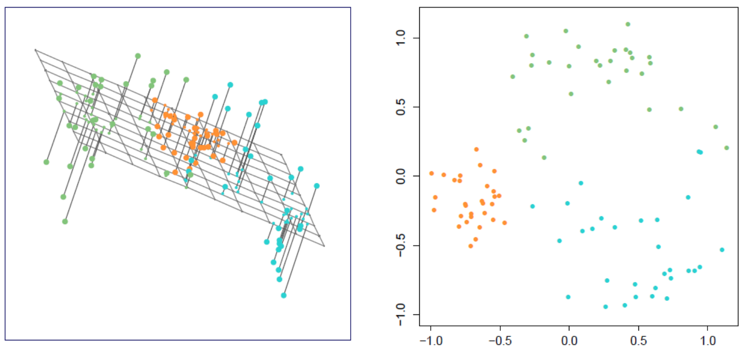


Figure 2.4: The figure on the left displays the input data. The figure on the right shows the result after applying PCA

the matrix $X \in \mathbb{R}^{n\times d}$ from the data:

$$X = \begin{bmatrix} — & x_1 & — \\ & \vdots & \\ — & x_n & — \end{bmatrix}$$

XX^T will be a **semi-definite positive** matrix and will therefore have $r \leq \min\{d, n\}$ eigenvalues and eigenvectors.

$$\text{Eigenvalues: } \lambda_1 \geq \lambda_2 \geq \ldots \lambda_r \geq 0$$
$$\text{Eigenvectors: } q_1, q_2, \ldots, q_r$$

As with the majority of the methods analysed, it is very important to normalise the variables. Otherwise, the variables with the highest values will have the greatest weight compared to the others. So, how do we know how many variables or dimensions to maintain?

There is no concrete answer to this question but, broadly speaking, we look at the variance to define the number of variables. The principal idea is to analyse the variance shown by the selected variables from the variance of the data as a whole. If it is *considerable*, we will retain these variables.[5]

As we see in Figure 2.5, after applying PCA to a dataset with a single variable or dimension, more than 60% of the total variance is expressed; two variables would cover more than 80%; and three or more variables will account for more than 90%. In view of this, we need to determine the number of variables to be considered, for which there are no set rules. The choice is subjective and made on a case-by-case basis, considering the following two points in particular: the new variables should comprise a *high* percentage variance and should not increase the percentage variance. In the example given, it is quite common to have two variables.

[5]The mathematical details of the variance can be read in section 10.2.3 of [17].

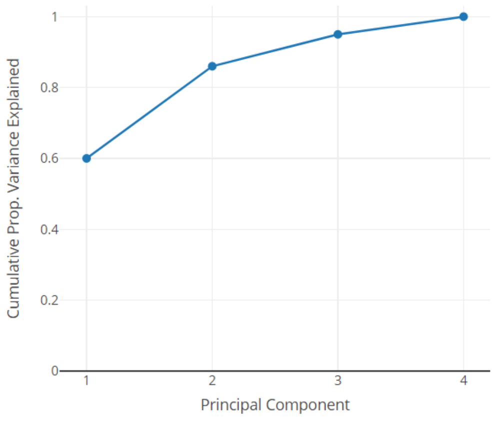


Figure 2.5: Percentages of cumulative variance obtained using different quantities of variables through PCA.

2.3 Matrix factorisation

Nowadays, a significant part of the market is online. We can buy or rent films, music, clothes, food, electronic devices, and more. The range of items that can be bought or rented is vast, making it impossible to see and analyse everything. This is why **recommendation systems** have proved especially important in recent years.

One of the methods used in these systems is **collaborative filtering**. In this case, we have *users* on the one hand and *items* on the other. The aim is to store the ratings given by users and learn from them. Thus, the algorithm will be capable of recommending new items by analysing the users' ratings.

The principal idea is: if two individuals assign a very similar rating for a series of products, they are likely to give other products a similar rating. In short, we are looking for a system for grouping users and items.

Matrix factorisation is an algorithm used for the **filtrado colaborativo** method.

The idea behind the algorithm

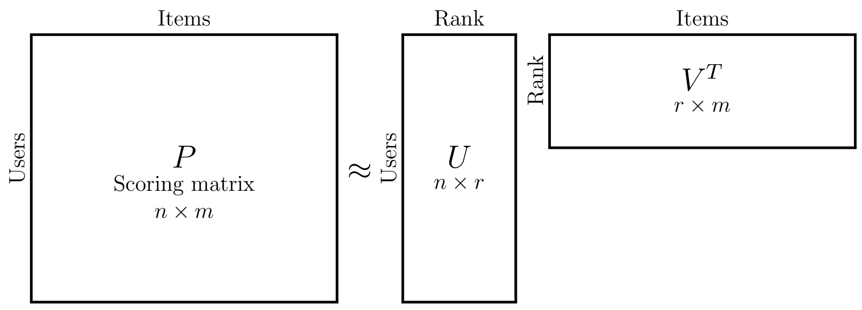


Figure 2.6: Matrix factorisation schema.

Put simply, this method creates a matrix P at the outset. The items are placed in the matrix columns, while the matrix P rows are the ratings. In other words, the element p_{ij} of the matrix P, will be the rating that user i has given the element j (from 1 to 5 or 1 to 10, etc.). In this matrix, most elements will remain empty since each user will evaluate only a small number of items, and the purpose of the algorithm is to make an accurate estimate of the missing values.

The single value decomposition[6] (SVD) is used for this. This gives the eigenvalues on which the matrix is *based.*

However, SVD cannot be used directly for recommendation systems because most of the values – around 99% of them – are missing from matrix P. We therefore need to obtain a schema similar to the one in Figure 2.6, where $p_{ij} \approx u_i^T \cdot v_j$ and the error to be minimised is:

$$\sum_{(i,j)\in\kappa} (p_{ij} - u_i^T \cdot v_j)^2 \tag{2.4}$$

The non-null components in matrix P will be in group κ. Furthermore, if we are considering using regularisation when calculating the error (further information on regularisation can be found in chapter 1 and it is defined in section 1.1.1), the error to be minimised will be:

$$\sum_{(i,j)\in\kappa} (p_{ij} - u_i^T \cdot v_j)^2 + \lambda(\|u_i\|^2 + \|v_i\|^2) \tag{2.5}$$

There are two main algorithms for minimising the error, *Stochastic Gradient Descent* and *Alternating Least Squares*[7].This paper will explain the main concepts behind alternating least squares.

- Firstly, the matrix V^T is started, randomly selecting the v_j vectors from the distribution $N(0, \lambda^{-1}I)$.
- Once the values in matrix V^T have been defined, the values of matrix U are updated so that the value for (2.5) is the least possible.
- Having updated U, it is left as defined and matrix V^T is updated, which minimises value (2.5).
- Steps 2 and 3 are repeated until convergence is achieved

Observations

The following should be taken into account in order to apply matrix factorisation:

- The matrix for rating P must contain almost all null values.

[6]The details of SVD can be read in most linear algebra books, among others in section 6.3 of [33].

[7]The details of the algorithm can be found in the article [20].

- The range of matrix P should be much less than its dimension.

- In order for the model to yield good results, the correlation between users should be generated and the lines of matrix P should thus be correlated.

- It is possible that several elements obtained from matrix $\widetilde{P} = U \cdot V^T$ might have *anomalous* values that are negative and too high (with a rating of 6.1 on a scale of 1 to 5, for example). To solve this, the results must be fitted to the evaluation system.

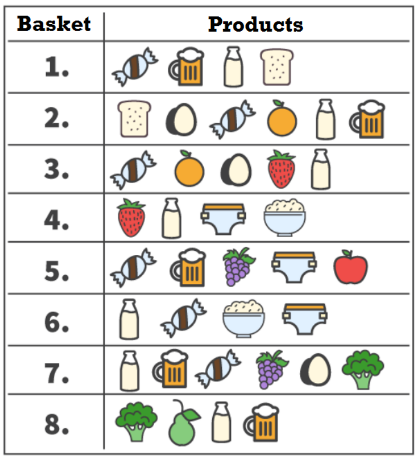
- In addition to *users* and *items*,other factors can be added such as search history, country, age, etc. Tensor factorisation[8] is used for such cases.

- The range or value r appearing in Figure 2.6 is not the range of the matrix P, since it cannot be calculated in the usual way. Different tests are therefore carried out for the different values of r (such as 10, 50, 100 and 200) and the most *desired* values are retained (those with the optimal error/computational cost ratio, for example).

2.4 Association analysis

Association analysis is a method generally used in business. Its aim is to identify products that generally appear in the same shopping basket. For example, we might suppose that whoever buys a personal computer will also buy, or be interested in buying, a case. It might therefore be useful to identify these relationships and try to offer consumers a package or collection of products.

As we will see below, the method uses basic, easily understandable mathematics and requires little pre-processing of data.



Basket	Products
1.	🍬 🍺 🥛 🍞
2.	🍞 🥚 🍬 🍊 🥛 🍺
3.	🍬 🍊 🥚 🍓 🥛
4.	🍓 🥛 🩲 🥣
5.	🍬 🍺 🍇 🩲 🍎
6.	🥛 🍬 🥣 🩲
7.	🥛 🍺 🍬 🍇 🥚 🥦
8.	🥦 🍐 🥛 🍺

Figure 2.7: An example of different shopping baskets.

[8]Section 19.4.2.1 of [30] can be consulted for more detail.

Figure 2.7 shows different shopping baskets with the aim of making the following two analysis:

- Association analysis, which looks at subsets comprising different products. Let us suppose that in the example from Figure 2.7 we wish to consider the link between the products {Beer, Milk}. We will calculate the probability of the basket as follows:

$$P(\text{The number of baskets they contain \{Beer, Milk\}}) = \frac{\text{\{Beer, Milk\}}}{\text{Total number of baskets}} = \frac{4}{8} = 0.5$$

- Association rules, which in a way measure the correlation between products and items. Let us suppose that in the example from Figure 2.7, if $A =$ {Beer, Milk} is bought, we are asked to measure the probability of buying $B =$ {Sweets}:

  $$P(B|A) = \frac{\text{The number of baskets they contain \{Beer, Milk, Sweets\}}}{\text{The number of baskets they contain \{Beer, Milk\}}} = \frac{3}{4} = 0.75$$

  We can also consider the relevance of the products $A =$ {Beer, Milk} when purchasing $B =$ {Sweets}:

  $$L(A,B) := \frac{P(B|A)}{P(B)} = \frac{3/4}{6/8} = 1$$

  In this case, the meaning of the 1 is as follows: there is the same probability that {Sweets} will be bought regardless of whether {Beer, Milk} are bought. If, for example, $L(A,B) = 2$, the meaning would change: it would be twice as likely that {Sweets} would be bought if {Beer, Milk} are purchased than if {Beer, Milk} were not.

These are the primary tools used for the analysis. However, where the quantity of data is vast, this analysis cannot be performed at the computational level. It will not be possible to analyse all of the item subsets, nor see the relationship between all of the items. As a solution, the **apriori algorithm** is used, the aim of which is to reduce the size of the giant tree being developed when the probability turns out to be too low.

Figure 2.8 shows a graph in which the complete tree has been constructed, but let us assume that the probability of basket $t \in [0,1]$ is greater than a value and that $P(AB) < t$. In this case, we will not only eliminate basket AB but also all baskets containing a combination of products from basket AB, since their probability will also be lower than t.

If we eliminate all of them, we will obtain the graph in Figure 2.9. In other words, a number of baskets will be eliminated a priori, which is the idea behind the **apriori algorithm**[9].

[9]For more information and details about the topic and the algorithm, section 14.2 of [15].

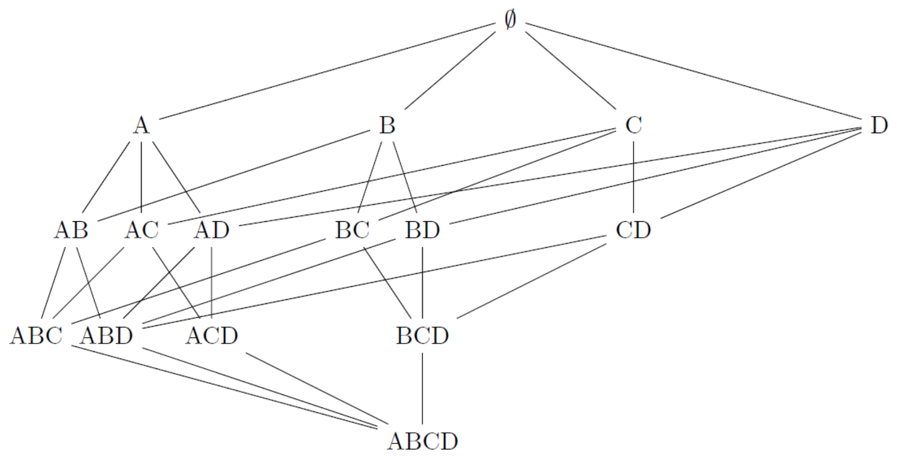


Figure 2.8: Graph of all the possible baskets with items A, B, C and D.

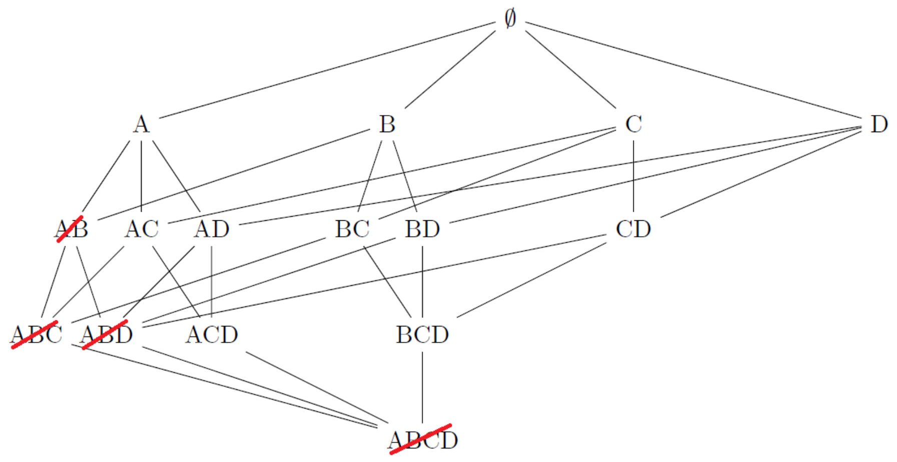


Figure 2.9: Schema of the apriori algorithm.

Chapter 3

Sequential data

Throughout this chapter we will discuss **sequential data**, which have a particular structure. This type of data is generally ordered in a specific manner, and this order has a special significance. If we look at the example of weather, the different measurements (precipitation, temperature, atmospheric pressure, etc.) are taken at different times, and these data are ordered chronologically in most cases. In this example, order and chronology are especially relevant because the measurements taken in November will not be the same as in June. At the same time, when making weather predictions, the weather recorded the previous day is relevant.

In the case of stock market information, the chronological order of share values is practically essential in order to conduct relevant analyses and make predictions. In chapter 3 we will therefore focus on the different analyses that can be performed with this type of data.

We will look at **Markov chains** and **Hidden Markov Models**. As we will see in the following subchapters, these methods allow us to form classifications or clusters. Unlike in chapters 1 and 2, however, the options that these methods give are much more general and diverse, which is why they were not included under supervised or unsupervised learning.

3.1 Markov chains

Markov chains owe their name to the Russian mathematician Andrey Markov and they refer to sequences of *memoryless* data. To put it another way, the result of each step in a sequence depends solely on the previous result[1]. If we consider the following data sequence:

$$\{x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n\}, \qquad x_i \in \mathbb{R}$$

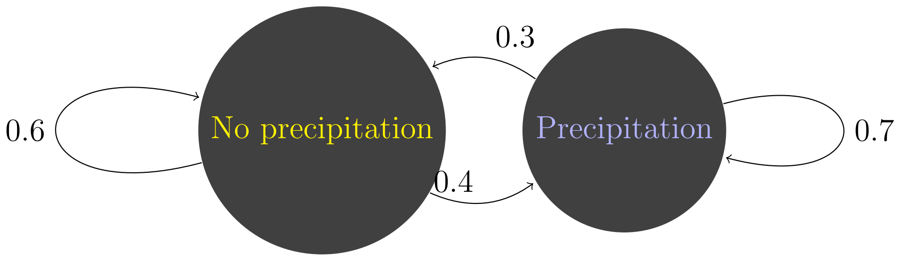
[1]The definition of Markov chains can be generalized, taking sequences that depend on the previous m elements. In that case, they would be called **Markov chains with memory of order** m.

where the value x_{i+1} depends solely on the value x_i, and $i \in \{1, \ldots, n-1\}$. The definition is generally presented in a probabilistic mathematical way:

$$P(x_{i+1}|x_i, \ldots, x_2, x_1) = P(x_{i+1}|x_i), \qquad \forall i \in \{1, \ldots, n-1\}$$

We will once again take the example of the weather. Making a considerable simplification, let us suppose that today's weather depends solely on yesterday's weather and that the probabilities are as follows:

$$P(\text{“No precipitation tomorrow”} \mid \text{“No precipitation today”) ”}) = 0.6$$
$$P(\text{“No precipitation tomorrow”} \mid \text{“Precipitation today”}) = 0.3$$
$$P(\text{“Precipitation tomorrow”} \mid \text{“No precipitation today”) ”}) = 0.4$$
$$P(\text{“Precipitation tomorrow”} \mid \text{“Precipitation today”}) = 0.7$$



Using these probabilities, we can create a **transition matrix** or **Markov matrix** as follows:

$$M = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{bmatrix}$$

The sum of the elements for each of the **transition matrix** lines will always be 1 given the properties of the probabilities, while there is a probability that element M_{ij}will go from state i to state j. In the above case, element M_{12} indicates that, knowing that yesterday was sunny, there is a probability that precipitation will occur today.

The transition matrix M is especially significant in Markov chains because it completely defines the chain. There are different ways of obtaining values from this matrix depending on each individual case. Elements M_{ij} are sometimes predefined, or the user should define them (they were predefined in the examples above). On other occasions, however, they should be estimated using the maximum likelihood method with respect to the input data.

Furthermore, it is an essential element for obtaining the (*stationary distribution*)[2], meaning that the following definition is very important for its applications:

$$M^{\infty} := \lim_{n \to \infty} M^n$$

[2]For more information about the stationary distribution and transition matrix: section 1 of [22].

Applications

Semi-supervised classification::[3]

This is useful in the cases shown in Figure 3.1, when only a few of the input points are classified. In these cases, we start with an unclassified point and jump from one point to another along a random line until reaching a classified point. They are then classified in the same group[4]. The probability of jumping from one point to another can also be defined in different ways, but in the following cases this is done according to distance: nearby points will have a greater probability compared to points further away.

Moreover, the points classified at the outset (the orange and blue points in Figure 3.1) will be the **absorbing points**, i.e. once they are reached they cannot be left. If x_{px} is an absorbing point:

$$P(x_{i+1} = x_{px} | x_i = x_{px}) = 1$$

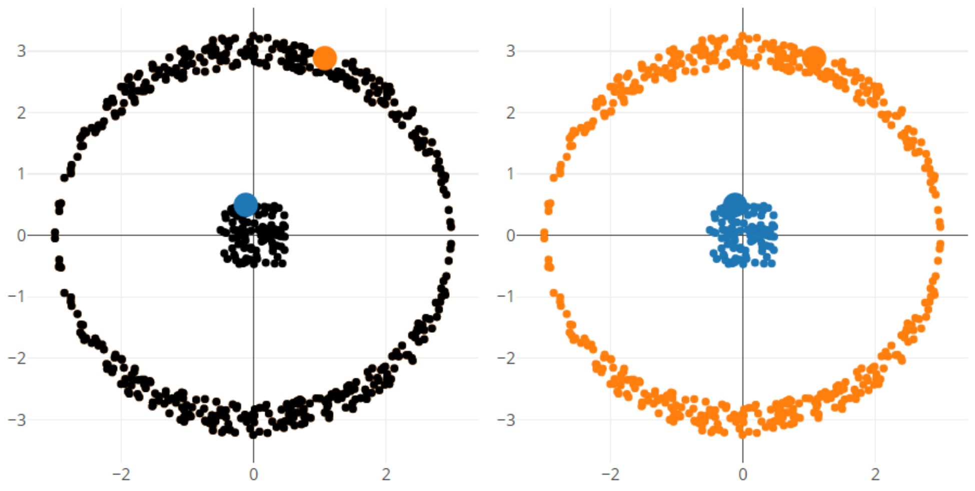


Figure 3.1: The figure on the left displays the input data. It shows two classified points, the blue point and the orange point, while the rest are unclassified. On the right, we can see the classification after applying the Markov chain.

Creating rankings:

Another possible application would be the creation of a ranking or list. For this, a transition matrix is developed to attain the objective artificially. We can, for example, make a Basque pelota ranking. For simplicity, let us suppose that a player is better than another if he wins the game, and the final result indicates *how many*

[3]In the following link you can see an animation explaining the method:
`https://www.datasciencecentral.com/profiles/blogs/a-semi-supervised-classification-algorithm-using-markov-chain-and`

[4]Instead of calculating using a random route, the classification of each point is calculated using matrix M^∞. For the classification of element j you have to look at row j of matrix M^∞ observing the column with the largest value.

times better he is. In other words, if player A beats player B, A would be better than B, and the difference in the final results would determine how much better.

Pelota players	A	B	C
A	X	22 - 18	22 - 11
B	18 - 22	X	22 - 20
C	11 - 22	20 - 22	X

Table 3.1: As we can see, in this example player A won both games, while C lost both.

With this definition, the movement of player B to player A would have a high probability in the transition matrix, whereas the probability of moving from player A to B would be low. Once the transition matrix M has been defined, matrix M^∞ is calculated. All of the values in matrix M^∞ will be equivalent. The value of column j will be the same as for pelota player j: the higher, the better.

In the case of pelota players, the ranking system will be devised as follows:

$$M_{ij} = \frac{\text{In the match between } i \text{ y } j \text{ the goals made by } j}{\text{In the matches of } i \text{ the goals that the losers have made } + 22}, \quad i \neq j$$

$$M_{ii} = 1 - \sum_{j \neq i} M_{ij}$$

The following transition matrix is obtained with this ranking system:

$$M = \begin{pmatrix} \frac{22}{18+11+22} & \frac{18}{18+11+22} & \frac{11}{18+11+22} \\ \frac{22}{18+20+22} & \frac{18}{18+20+22} & \frac{20}{18+20+22} \\ \frac{22}{11+20+22} & \frac{22}{11+20+22} & \frac{9}{11+20+22} \end{pmatrix} = \begin{pmatrix} \frac{22}{51} & \frac{18}{51} & \frac{11}{51} \\ \frac{22}{60} & \frac{18}{60} & \frac{20}{60} \\ \frac{22}{53} & \frac{22}{53} & \frac{9}{53} \end{pmatrix}$$

and with this we achieve matrix M^∞ as defined previously:

$$M^\infty = \begin{pmatrix} 0.4047482 & 0.3496906 & 0.2455612 \\ 0.4047482 & 0.3496906 & 0.2455612 \\ 0.4047482 & 0.3496906 & 0.2455612 \end{pmatrix}$$

Thus, according to the ranking, player A was the best, while C had the worst results.

3.2 Hidden Markov Models

Unfortunately, there will be times when it will impossible to conduct analyses with the data directly. Let us suppose, for example, that we want to see when methane rain falls on the moon Titan[5]. We will assume that the analysis is conducted using an infrared telescope and that the research results are as follows:

$$P(\text{“No precipitation”} \mid \text{Level of infrared} < T_0) = 0.2$$
$$P(\text{“Precipitation”} \mid \text{Level of infrared} < T_0) = 0.8$$
$$P(\text{“No precipitation”} \mid \text{Level of infrared} \geq T_0) = 0.9$$
$$P(\text{“Precipitation”} \mid \text{Level of infrared} \geq T_0) = 0.1$$

where T_0 represents a known value obtained from the research. In this case, we cannot know the specific result using the tools we have; only the level of infrared. Let us also suppose that further research found the following probabilities:

$$P(\text{“No precipitation tomorrow”} \mid \text{“No precipitation today”}) = 0.6$$
$$P(\text{“No precipitation tomorrow”} \mid \text{“Precipitation today”}) = 0.3$$
$$P(\text{“Precipitation tomorrow”} \mid \text{“No precipitation today”}) = 0.4$$
$$P(\text{“Precipitation tomorrow”} \mid \text{“Precipitation today”}) = 0.7$$

The goal is to determine changes to the weather expected on Titan and to attempt to predict the weather based on the data collected (by measuring the level of infrared):

$$\{t_1, t_2, t_3, \ldots, t_n\} \quad n \text{ infrared level measurements}$$

In this case, there will be two matrices: **the transition matrix**:

$$A = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{bmatrix}$$

and the **emission matrix**:

$$B = \begin{bmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{bmatrix}$$

The initial distribution of precipitation or no precipitation is:

$$\pi = [0.5 \quad 0.5]$$

In this example, the matrices A, B and π are regarded as known, but this will not be true in most cases and the matrices would have to be estimated. Furthermore, in this case the hidden Markov model is a discrete model, i.e. the *states* it gives are discrete (precipitation or no rainfall). Three main estimation problems arise with discrete models:

- Obtaining the estimated state for each t_i when the matrix $\{A, B, \pi\}$ is for a discrete *HMM* and the input data series is $\{t_1, t_2, \ldots, t_n\}$. This problem can be solved using the **forward-backward algorithm**.

[5]For more information about the Titan satellite:
`https://www.space.com/15257-titan-saturn-largest-moon-facts-discovery-sdcmp.html`

- Obtaining the most likely *successive state* $\{s_1, s_2, \ldots, s_n\}$ when the matrix $\{A, B, \pi\}$ is for a discrete *HMM* and the input data series is $\{t_1, t_2, \ldots, t_n\}$. This problem can be solved using the **Viterbi algorithm**.

- Estimating the matrices $\{A, B, \pi\}$ with the analysed sequence $\{t_1, t_2, \ldots, t_n\}$. For this we use the **maximum likelihood** method.

After *HMM*, there is an entire area of mathematics and detailed theory that we will not go into here as it is not the purpose of this paper. Nonetheless, for further reading we recommend article [29] and textbook [4] for their highly useful insights in this area. They even include continous *HMM*, i.e. states without discrete *states*, which are even more complex but which have more interesting applications, such as intelligent cars.

Chapter 4

Artificial neural networks

4.1 Historical background

Artificial neural networks were first studied in the 1940s when various papers were published on the topic. They managed to make simple arithmetic calculations but with major limitations. The main advances took place in the 1950s and 1960s, thanks to the *Perceptron*algorithm which seeks to imitate the function of a neuron and which is regarded as the precursor to neural networks. The main development brought about by the algorithm was its ability to select the most suitable parameters by itself. As we can see in Figure 4.1, let us suppose that the input data are in a binary class (shown in the figure in green or red) and the goal is to classify them using a hyperplane. For this, the algorithm *manages* the weights $W = (w_1, \ldots, w_n)$ until the objective is achieved for each iteration (where this is possible).

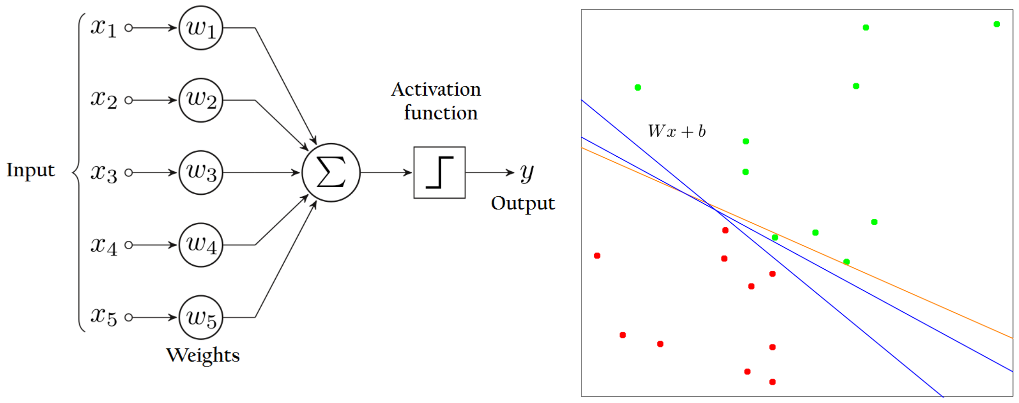


Figure 4.1: On the left, we see the *Perceptron* schema, whereas on the right we have the algorithm's process up until it selects the appropriate hyperplane.

In spite of these advances, research into artificial neural networks came to a halt during the 1960s and 1970s because the capacity of computers at the time was low and because there were no suitable algorithms for training the networks. This changed in 1986 thanks to the **backpropagation** algorithm, which provides neural networks with the capacity to *learn* or select the appropriate weights automatically. Since then, the field of neural networks has experienced significant growth and development (and still is), as it has been shown to have limitless capacities.

4.2 Main concepts

The idea behind artificial neural networks is to imitate or emulate processes within our brains, particularly when generating the scheme. Figure 4.2 describes the process in a simple way. The circles are used to indicate neurons and the arrows show the connection between them. In this simple case, each neuron indicates total dependency on the neurons to their left, depending on which ones will activate or not.

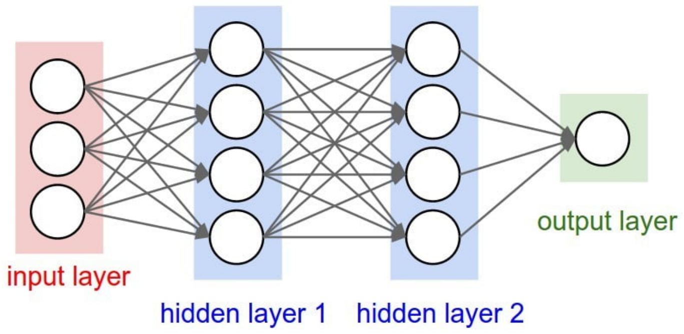


Figure 4.2: Basic schema of an artificial neuron.

Let us suppose that, building on Figure 4.2, we have to analyse the case below of data concerning apartments on sale in San Sebastián over the last 50 years:

- Sale price
- Number of square metres (m^2)
- Age of the apartment in years
- A binary variable indicating whether or not the apartment was sold in two months

Let us suppose that the goal is to create a network capable of indicating whether or not it was sold in two months using the three input variables (it can be regarded as a classification method in chapter 1).

The neural network process comprises the following steps:

- Go from the input data $X \in \mathbb{R}^{1\times3}$ (**input layer**: price, m^2 and number of years) to the next layer (**hidden layer 1**) through matrix multiplication:

$$X \cdot M_1 = H_1' \text{ where } M_1 \in \mathbb{R}^{3\times4} \text{ and } H_1' \in \mathbb{R}^{1\times4}$$

- Once we find H_1', we will apply an **activation function**[1] so that the numbers H_1' are converted to 0 or 1 (to activate the neurons or leave them deactivated):

$$f(H_1') = H_1 \text{ where } H_1 \in \mathbb{R}^{1\times 4}$$

- Transfer H_1 from **hidden layer 1** to the next layer (**hidden layer 2**) through matrix multiplication:

$$H_1 \cdot M_2 = H_2' \text{ where } M_2 \in \mathbb{R}^{4\times 4} \text{ and } H_2' \in \mathbb{R}^{1\times 4}$$

- Once we find H_2', we will apply an **activation function** so that the numbers H_2' are converted to 0 or 1 (to activate the neurons or leave them deactivated):

$$g(H_2') = H_2 \text{ where } H_2 \in \mathbb{R}^{1\times 4}$$

- Transfer H_2 from **hidden layer 2** to the next layer (**output layer**) through matrix multiplication:

$$H_2 \cdot M_3 = y' \text{ where } M_3 \in \mathbb{R}^{4\times 1} \text{ and } y' \in \mathbb{R}$$

- Finally, once we obtain y', we apply an **activation function** so that the number y' becomes 0 or 1 (whether or not the apartment was sold in two months).

$$h(y') = y \text{ where } y \in \{0,1\}$$

4.3 Learning process

In the process we have just explained, the matrices M_1, M_2 and M_3 have appeared but the origin of their values has not been specified. The main objective of the network is to determine their values since the appropriate values for these matrices will determine whether or not the output value is satisfactory. As we mentioned in the introduction to this section, the method or algorithm used to select the appropriate values for these M_i matrices is **backpropagation**. We use derivatives to minimise the error.

We will not explain the mathematics or the details behind the method since it is not the purpose of this manual. In any case, further information on the topic can be found in practically any book on artificial neural networks, particularly [13, 21]. Furthermore, Andrej Karpathy, the director of artificial intelligence at Tesla, has a highly interesting video on the internet explaining the backpropagation method:

`https://www.youtube.com/watch?v=59Hbtz7XgjM`

[1]The most used activation functions: sigmoid function, arc tangent function and ReLU function. `https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0`

Thus, neural network learning involves continuously repeating the process shown in Figure 4.3. The input data are entered and the matrices M_1, M_2 and M_3 give the output result. If the result is correct, the values of the M_i matrices are not changed. If the result is incorrect, on the other hand, the values of the M_i matrices will be revised using backpropagation.

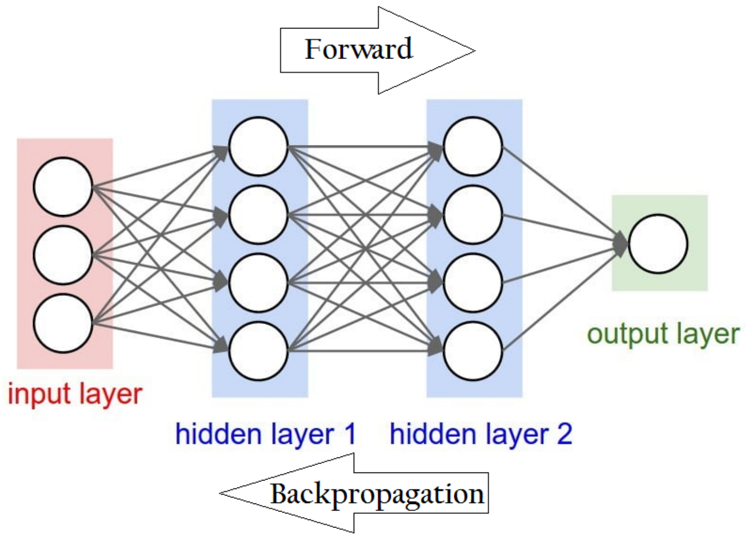


Figure 4.3: Basic schema of an artificial neuron.

4.4 Advantages and disadvantages

Artificial neural networks have played a key role in recent years as new applications are always appearing. No limits have been set for the moment. They are capable of creating classification and regression models for supervised learning, and of reducing the dimensionality of the data seen in unsupervised learning (the autoencoder). Moreover, they are capable of performing highly complex tasks for most methods, including image classification, sound identification, translation, etc.

There are a series of tangible drawbacks, however. On the one hand, few theories have been developed about them and most of the applications and results have been produced using different tests. There are no theories for selecting the number of layers or establishing the number of neurons. We can only find recommendations as to where they have worked in previous cases. On the other hand, the models obtained are *black boxes*, since in many cases it is impossible to know the details of the model. To put it another way, although they can provide satisfactory results, most of the time we will not know how the classifications and estimates have been made.

Chapter 5

Ensemble methods

The goal of ensemble methods is to develop diverse models which can be combined to generate better results. In the case of decision trees, for example, these methods are often used to improve results. The decision tree algorithm is generally simple and streamlined, and can be leveraged to develop many different trees. Better and more solid results are thus obtained by combining them[1].

In general, these methods are divided into two families:

- *Averaging methods* or bagging methods, where different estimators are developed, first independently and then after calculating the mean. When finding the average, the new estimator will generally be better since it reduces bias.
- With boosting methods, on the other hand, the estimators are generated sequentially and the bias of the estimator is improved with each step.

5.1 Bagging methods

As we said in the introduction to this chapter, this method creates a series of models with input data and then generates one using the average, which will have enhanced properties. We outline the main concepts behind the algorithm below:

- Firstly, the values of n and M are selected. M will represent the number of samples to be created, while n will be the number of elements to be used to generate the models.
- The first step will be to collect the samples. M samples will be created and n elements of input data will be selected for each. These elements will be selected at random, and will be different for each sample.
- Once M samples have been generated, a model will be developed for each sample with whichever method we wish to use (regression or classification).

[1]In *Netflix Competition* [7] for example, to make a robust collaborative filtering algorithm the Ensembe methods were used.

- The final model will be the average of the preceding models. In the case of classification, the class of a point will be the class that gave the most positive results with the models generated rather than the average.

Nonetheless, the samples obtained through this method will often show a correlation, and this will exert a negative impact on the quality of the final model. To eliminate or reduce this correlation, the random tree method is used.

Random trees

The difference with respect to the algorithm that we have seen lies in the selection of the variables. If the input data have k variables, only d variables in the sample will be selected at random when choosing the sample. Generally, $d \approx \sqrt{k}$ is selected. This reduces correlation among samples, thereby producing better results.

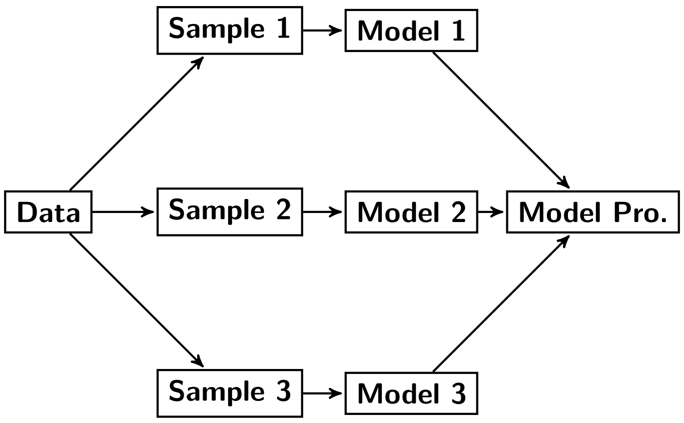


Figure 5.1: Schema of the bagging method: samples of identical sizes are selected from the input data, models of these samples are generated and, finally, an average sample of these models is calculated.

5.2 Boosting methods

This method essentially seeks to improve them sequentially, based on simple models. It is used only for classification methods. This improvement is achieved by giving greater weight to the data poorly classified by the model. We will outline the main concepts behind the *AdaBoost* (Adaptive Boosting) algorithm[2] below:

- As with the bagging method, the values of n and M are selected to start with. M will represent the number of samples to be created, while n will be the number of elements to be used to generate the models.

[2]The following video shows the operation of the algorithm graphically: `https://www.coursera.org/learn/ml-classification/lecture/um0cX/example-of-adaboost-in-action`

- A simple random sample of size n is then obtained. This step is only carried out the first time, since in subsequent steps several elements from this sample will have a higher probability when the sample is collected, depending on the weight assigned to them.
- A model will be allocated to this sample using the classification method.
- This model will include poorly classified elements and give them a greater weight so that they have a higher probability of appearing in the next sample.
- A sample of size n is obtained with the new probability distribution, and the same process is repeated M times.
- Finally, after having repeated the process M times, the final model will be completed using the weighted average of the M models (with the weights obtained from the algorithm).

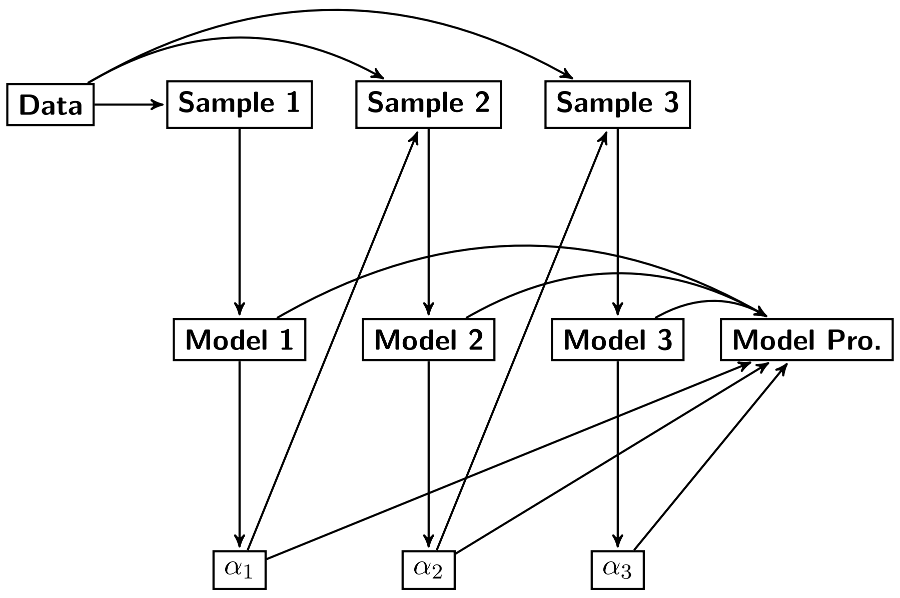


Figure 5.2: Schema of the AdaBoost method. The values α_i correspond to the weight obtained from the models, and will be used to create the next sample and the final model.

This algorithm is valid for any classification method and, as we pointed out, it serves to improve their results and minimise the error. However, they are generally applied to simple methods such as decision trees since they have a lower computational cost.

Part II

Working with data

Introduction

This section will explain the work carried out on clustering and the results obtained – the main objective of this grant. The daily prices of the hotels and guesthouses in the Basque Country have been gathered from one hotel offer website. Different methods of web scraping were used to collect data and, as we began to understand the structure of the website, the quality of the data also improved.

To set the daily prices, a total of 120 searches on the website were conducted every day. The web scraping program gathers the prices for each hotel and guesthouse in the Basque Country for the next 120 days. For example, on 1 January 2018, the program requested the prices for every day between 1 January and 1 May for every hotel and guesthouse. This is why, in the best case scenario, every hotel establishment will have a total of 120 prices, one for each day.

This is practically impossible, however, given the numerous drawbacks. On the one hand, not all of the establishments offer the option of booking via the website. For this paper, for example, the hotel coverage was approximately 75%. On the other hand, the data collection program might experience a range of problems (an unexpected error, changes in the website structure, fully booked/closed hotels, etc.), in addition to any issues with the website itself.

After sending these 120 requests comes the process of choosing the meaningful price for each day. This analysis was left to a research project by the University of the Basque Country and, in the meantime, it was decided for the purposes of this work to consider the median of all the prices, since this is a central measure that excludes anomalous values.

These data were merged with the Eustat *Survey on Tourist Establishments* in order to have additional information on hotels and guesthouses.

After these preliminary steps, we decided to use the programming language `R` [28] to process the data. There are several reasons for this choice: it is a free programming language, it has gained importance in the world of machine learning in recent years (together with the programming language Python) and it is designed for statistics.

This work process is broken down into three major parts:

- Outliers in time series
- Imputation in time series
- Clustering of time series

The main `R` packages used to present the results were `shiny` [6], `plotly` [32] y `leaflet` [8].

Chapter 6

Outliers in time series

When conducting different analyses, the results are highly dependent on the data, as is logical. Therefore, if the data gathering process was faulty or if there are errors, the reliability of both the results and the conclusions derived from the analysis may prove dubious. This case is no exception; we have tested different methods to detect anomalous values and outliers in our time series, creating an algorithm suited to our needs.

At the outset, we tested the `tsoutliers` [9]] and `outliers` [19] packages, and immediately found that they were of no use for our case (at least not directly). Using the functions offered by these packages, the series are analysed individually and the high prices for special days are detected as outliers. In the case of `tsoutliers`, the effects of the calendar can be determined (weekends, public holidays, Easter, etc.); however, this is sometimes not enough, since there are also special events.

We look at Figure 6.1, we will see that there is an anomalous value on 11 November. It was a Saturday and there was no special public holiday in the Basque Country. Nonetheless, since there was a similar pattern in among most of the hotels and guesthouses, we looked to see whether there was a special event in San Sebastián. It turns out that it was the day of the *Behobia-San Sebastián half marathon.*

We thus concluded that we would need to compare our time series with each other. The difference between an anomaly or outlier is whether or not the anomalous value occurs for other hotels and guesthouses.

6.1 Constructing an algorithm to detect outliers

As we asserted at the start of this section, to detect outliers we need to compare the various hotels and guesthouses. The main idea is, firstly, to detect the anomalous values in the time series and then decide whether they are outliers or anomalies by comparing the time series.

The process for this analysis will be divided into four parts.

Organisation of the time series by month

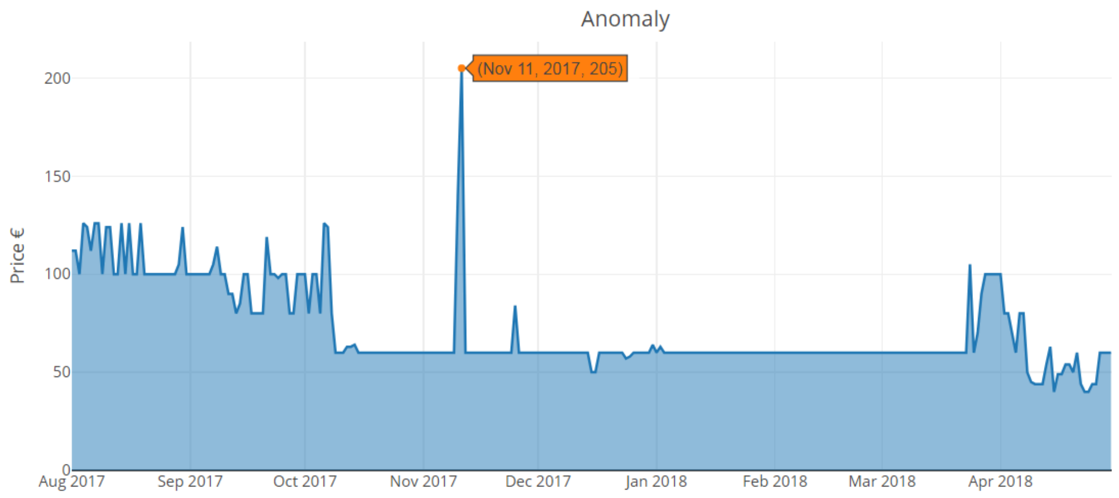


Figure 6.1: Time series of prices of a guesthouse in Donostia/San Sebastián. We can see an anomaly on 11 November, the weekend of the *Behobia-San Sebastián half marathon*.

With the aim of cleaning up data gathered automatically for the future, we decided to analyse the outliers in monthly time series. Furthermore, there is no use in comparing different months because every season has different trends.

Eliminating minimum outliers

Once the data have been grouped by month, the next step is to eliminate the minimum outliers. A very simple concept is used for this: the prices for each hotel and guesthouse will be tested by month and the result will help us decide whether to eliminate the value. The test employed for detection is based on the Grubbs method [14], which is included in the `outliers` package. This method analyses the outliers by group and eliminates the time factor, i.e. the day for any given price does not matter.

In the case of minimum outliers, no pattern is produced, unlike with maximum outliers. In other words, the prices are not all accumulated in a few days. The lowest prices are more common than the highest, or, to put it another way, the prices are generally stable throughout the entire month, and that stability is often broken by the maximum prices.

It is worth mentioning that the Grubbs test returns a series of statistics and an associated value p. One of the tasks will be to define this value p in order to determine an outlier. For this, we visually analysed the minimum outliers for the first five months, and we retained the value of p that best classifies them.

Finally, the values detected by the algorithm were eliminated and the minimum price for the month over those days was then assigned to each hotel and guesthouse.

Changing the price scale

Having finished analysing the minimums, the same should be done with the maximums. As we mentioned at the start of this section, the comparison for this task should be made among the establishments. The scale used is very important for the comparison, since a luxury hotel cannot be compared with an inland hotel with one star, at least not in absolute price terms. The price scale was thus changed as follows:

$$z_i = \frac{x_i}{\min(x)} \cdot 100 \tag{6.1}$$

donde

- x: time series of prices (by month)
- x_i: price of day i
- z_i: rescaled price of day i

The reason for changing the price scale is to measure the increase maintained by a hotel or guesthouse throughout the month. As an example, we will rescale the following vector:

$$(70, 105, 70, 140, 210) \longrightarrow (100, 150, 100, 200, 300)$$

those with the value 100 indicate the minimum price, whereas the other values show the percentage increase, i.e. the value 200 indicates that the price has doubled (with respect to the minimum price) and the value 150 will indicate that the value has been multiplied by 1.5 ($70 \cdot 1.5 = 105$). By changing the scale, we have managed to compare the price changes.

Eliminating maximum outliers

Once the pre-processing of the data is complete, two analyses will be conducted to detect the maximum outliers: one **monthly** and one **daily**. We will use the same method we followed with the minimums, namely detecting the outliers for each establishment with the Grubbs method. Figure 6.2 shows a real case, in which we can see four points classified as outliers, following the Grubbs test. Looking at the graph, 3, 4 and 25 November do not appear to be outliers; however, in this special case they are classified as such given that the other prices remained constant.

After this first step, the second will be to analyse the **daily price**. It will also be subject to the Grubbs test, but on a daily basis and comparing it to other hotels and guesthouses. Taking the month of November as an example again, we will see that a total of 30 Grubbs tests are performed, one for each day. The prices of the different hotel establishments are considered for each day (prices rescaled using formula (6.1) and the test is applied to that group. We can see the outliers obtained from the results for November in Figure 6.3.

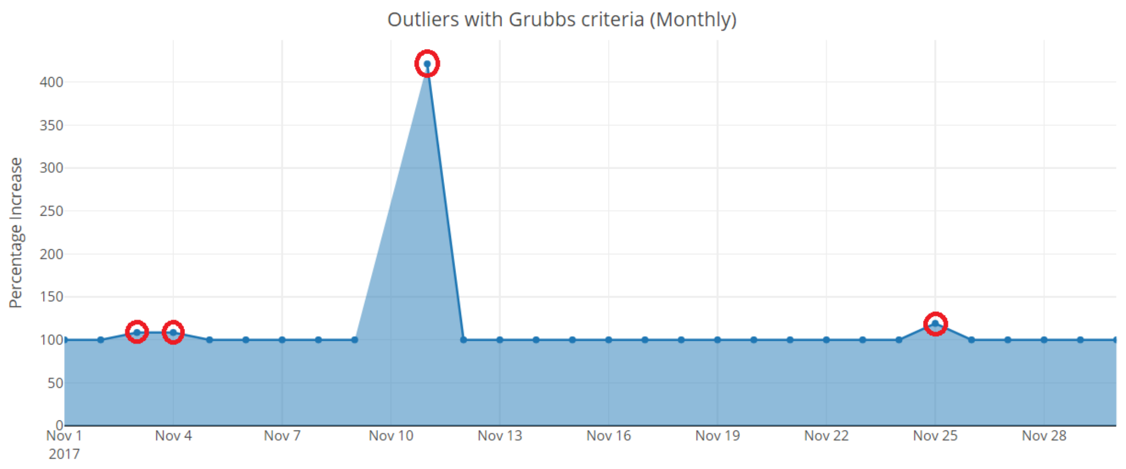


Figure 6.2: In the graph we can see four outliers, highlighted in red, which were determined by applying the Grubbs test to a hotel.

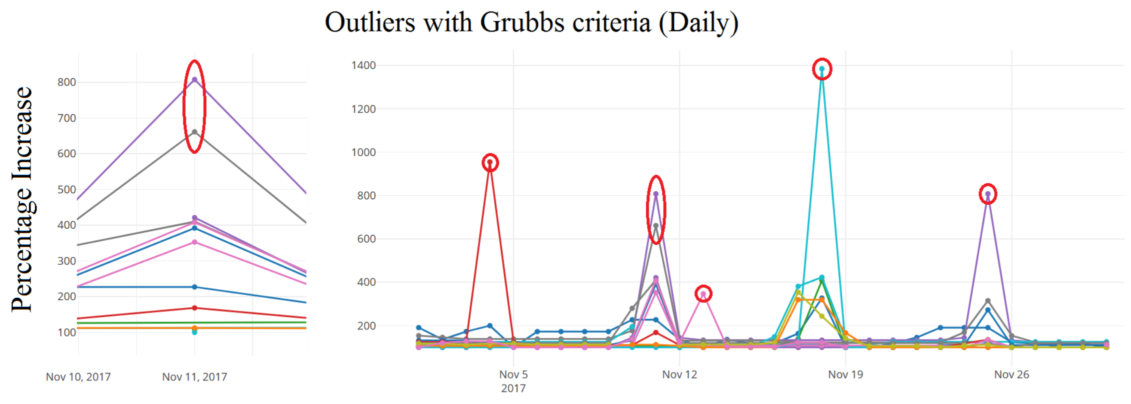


Figure 6.3: The outliers for every day in November are highlighted in red. In the graph on the left, we can see that 11 November and the two points considered outliers after conducting the test are highlighted with a red circle.

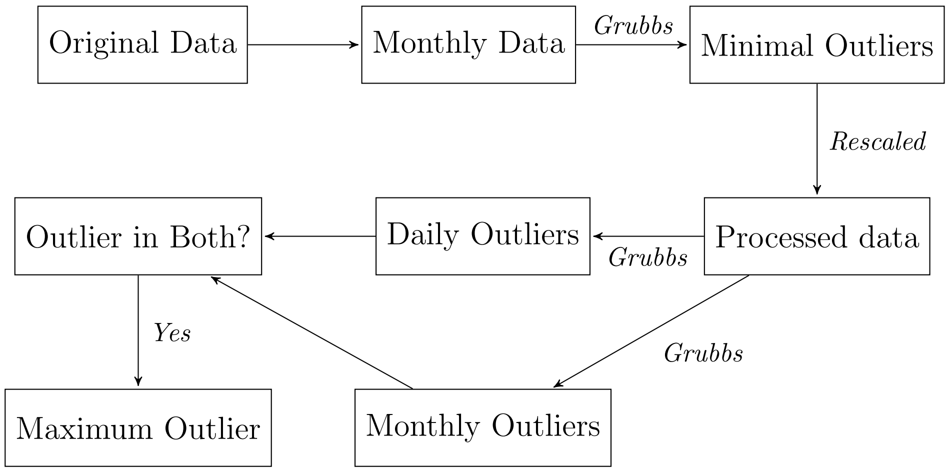
It should be borne in mind that, when conducting the analysis by day, the hotels and guesthouses were grouped by province. This avoids comparing an anomalous value that might occur in Vitoria-Gasteiz on 11 November with anomalous values in Donostia, since each province and capital city have their own trends.

After these two analyses or steps, the following decision is made:

- If a certain value is considered an outlier in both analyses $\Longrightarrow$ **OUTLIER**
- Otherwise $\Longrightarrow$ **NOT AN OUTLIER**

The example given will not be considered an outlier or unique value since, as we can see, most of the establishments sharply increased their prices on 11 November, meaning that the two conditions were not met. As in the case of the minimums, to define the value p for the Grubbs test, we visually identified the outliers for a total of 5 months and selected the value that best classifies them. The eliminated outliers were assigned the 3rd quartile of the monthly price.

The diagram below shows a summary of the steps in the algorithm:



Chapter 7

Imputation in time series

Time series sometimes have missing values, for various reasons. These missing values are sometimes interesting since they offer additional information. At other times, however, they may pose a problem and we might want to recover them.

In our case, missing values may have a different meaning, such as the hotel might be closed or fully booked or there were errors in gathering the data, for example. We can easily identify cases when they were closed, since we receive this information from surveys. When they were fully booked, or where there were problems, data imputation is a possible solution.

For the imputation of time series, the `R`-k [28] package `imputeTS` [26] contains several methods to choose from depending on our needs. In our particular case, the data have clear periodicity, i.e. the prices always increase on Fridays and Saturdays. In cases with tangible periodicity, the `imputeTS` documentation recommends using the following functions, as they give the best results in most cases: `na.seasplit`, `na.seadec` y `na.kalman`.

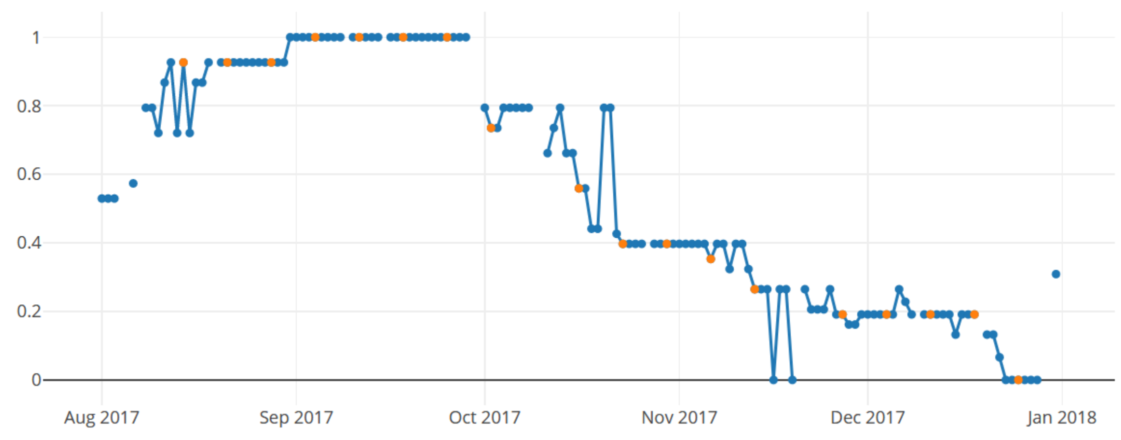
7.1 na.seasplit

This function comprises a series of steps in the imputation process. We need to tell the function the periodicity of the time series entered. Our case has weekly periodicity, meaning that the input function would be:

```
ts(x, frequency = 7)
```

where `x` represents our input data, namely the prices for a specific hotel. Firstly, the function will extract 7 time series from the original time series, one for each day of the week (see the example in Figure 7.1).

Once we have divided our time series into seven, we will apply the imputation method to each (there are several options in the `imputeTS` package: interpolation, ARIMA, basic structural model, mean, etc.).



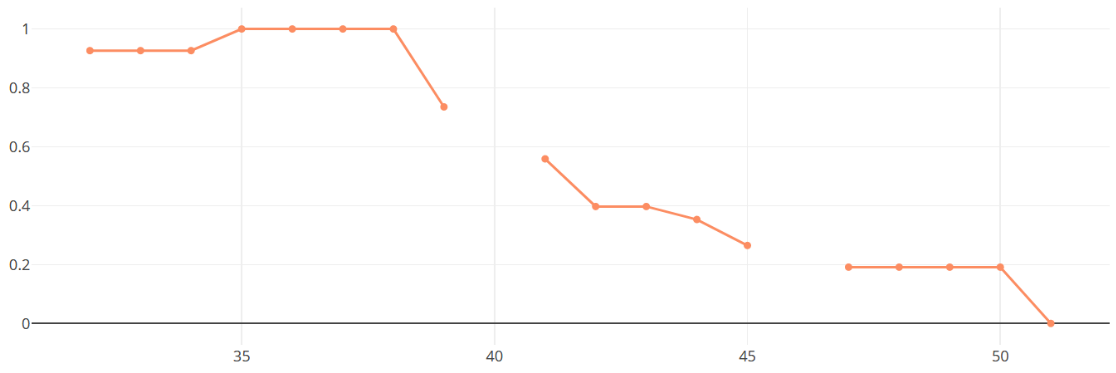


Figure 7.1: The first figure is an example of a time series with missing values. The orange points show Mondays. In the figure below, we only see the data for Mondays, according to the week of the year.

7.2 na.seadec

This method also comprises several different steps, and the periodicity has to be specified in order to exploit its advantages. In our case:

```
ts(x, frequency = 7)
```

Firstly, the series is imputed simply by using linear interpolation, with the function `na.interpolation`. Once the missing pieces of data have been filled in, the periodicity of the time series is eliminated by using the function `stl`[1]. Once the periodicity has been worked out, the remaining data are imputed using the method of our choice (there are several options in the `imputeTS` package: interpolation, ARIMA, basic structural model, mean, etc.) Finally, after imputation, the periodicity of the series is added.

[1]The `stl` function is integrated in `R`. It decomposes the time series into: *trend*, *seasonal part* and *rest*.

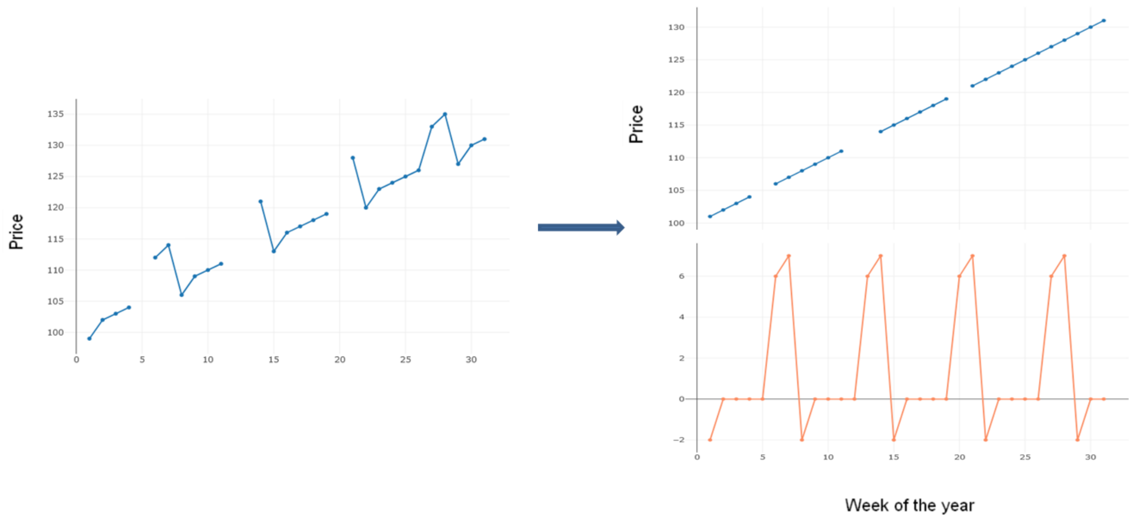


Figure 7.2: The time series to be imputed is shown in the figure on the left. On the right, we see the time series after applying the function `na.interpolation` and after eliminating the periodicity.

7.3 na.kalman

This function imputes the missing values through modelling. We can enter different models of the time series as required. In any case, this function includes two models: `auto.arima`[2] and `StructTS`[3]. The first is found in the `forecast` package [16], while the second is included in the `R` programming language.

Let us suppose that, to simplify things, we have obtained the following model:

$$y_t = a \cdot y_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N(0, \sigma)$$

i.e. each of the points in our time series is dependent only on the immediately preceding point. Thus, the function `na.kalman` draws the model using the **Kalman filter**[4]. Basically, the filter is able to select the *appropriate* ε_t noise when choosing the y_t values.

7.4 Selecting the imputation function

Once we have analysed and understood the functions recommended by the `imputeTS`documentation, we must select the one that best suits our needs. For this we will take all of our **complete** time series, i.e. around 150 time series that are missing 5 values or fewer between August 2017 and December 2017, on which we will test the methods.

[2]For more information about ARIMA models, section 3 of [31].
[3]For more information about *Basic Structural Model* (BSM) and *State-Space models*, section 6 of [31].
[4]For more information about **Kalman filter**, section 6.2 of [31].

We will take 15-20 points at random, giving more weight to weekends (since there are more missing values at weekends – see Figure 7.3). They will then be imputed with different methods. The mean squared error will be used to calculate the resulting error.

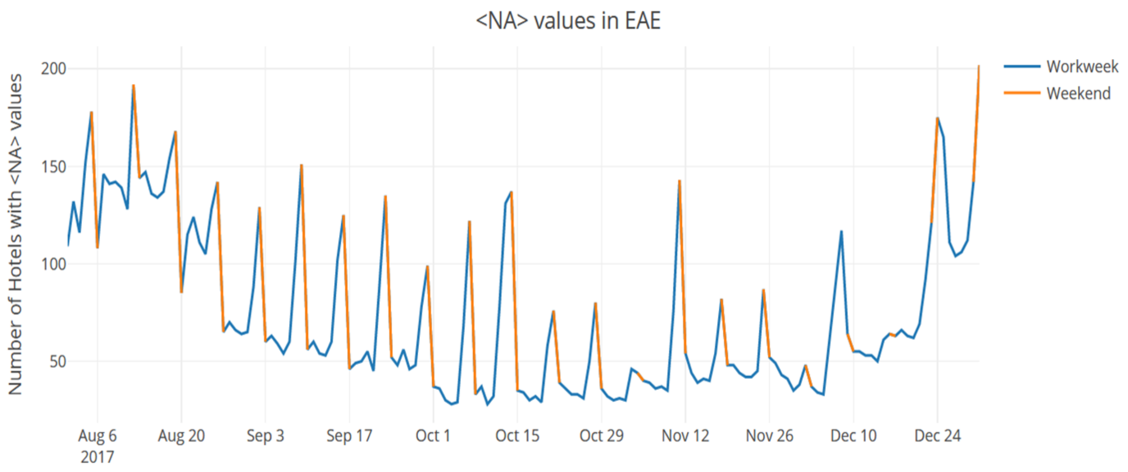


Figure 7.3: As we can see in this figure, the missing values between August 2017 and December 2017 are concentrated at weekends.

After repeating the test a total of 20 times, we arrive at Figure 7.4. The same is done with the minimum prices from the website[5] (knowing that they have a more prominent periodicity), the results of which are shown in Figure 7.6.

In view of the results, we opted for the `kalman` algorithm within the `na.seadec` function, i.e.

```
na.seadec(ts(x, frequency = 7), algorithm = "kalman")
```

[5]Remember that when calculating the daily prices, prices have been collected for 120 different days, keeping the median of those prices for the analysis. When choosing the imputation method, the minimum of these prices has also been taken into account, since they presented a greater seasonality.

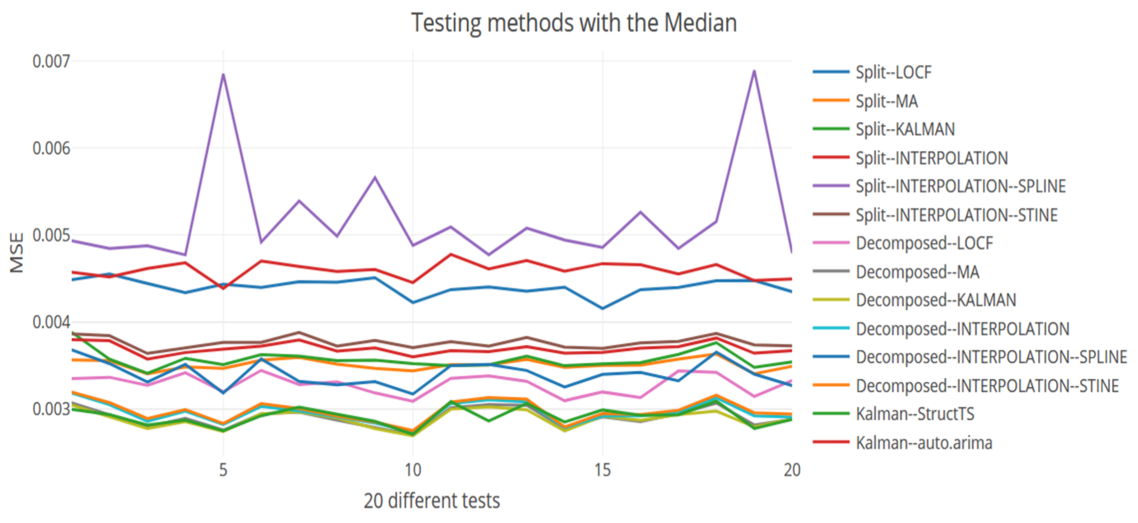


Figure 7.4: Imputation error resulting from different methods after 20 separate tests. `Split=na.seasplit`, `Decomposed=na.seadec` and `Kalman=na.kalman`. In the figure below, we see the methods that generate the fewest errors.

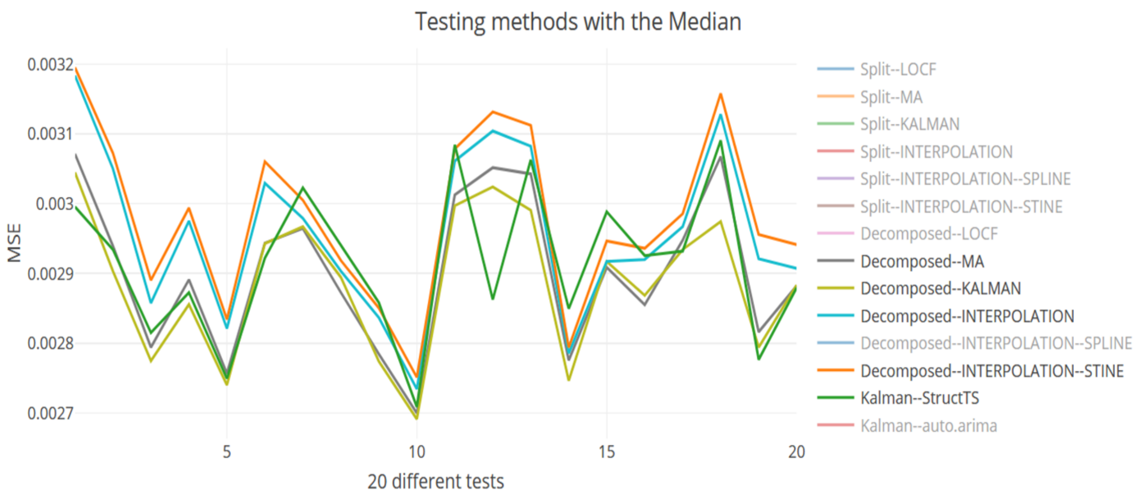


Figure 7.5: Methods with the fewest errors, after performing 20 separate tests.

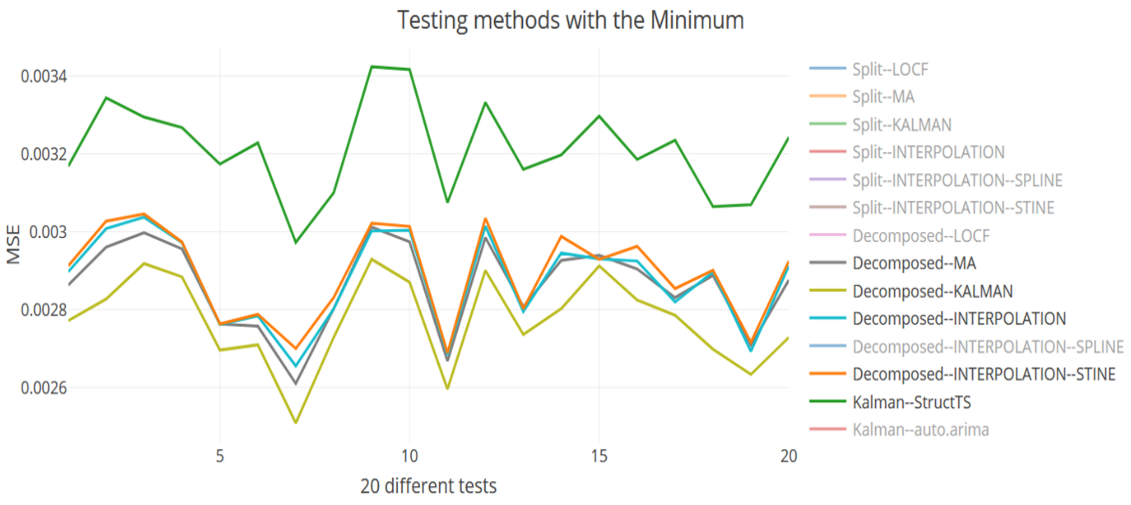


Figure 7.6: Error obtained after 20 separate tests using the minimum prices.

Chapter 8

Clustering of time series

Clustering the time series of prices is one of the main objectives of the grant. We used the following `R` [28] packages for this: `TSdist` [25] and `TSclust` [24]. Different distances between the time series are displayed in both packages in order to select the one that best suits the user's needs in any given case. For our analysis, the fact that the price fluctuations occur on the same day is given paramount consideration, since the intention is not to compare a hotel that had high prices in August with another with high prices in October. Therefore, distances as DTW [12] (Dynamic Time Warping) were eliminated for the analysis.

The main distances tried were L_p (Manhattan and Euclidean) and those derived from the time series characteristics (distances by correlation/autocorrelation, distances by Fourier coefficients, distances based on the ARMA/ARIMA models, etc.) The clusters are similar for most of them, so we decided to take the distance that is easiest to understand: **Euclidean distance**.

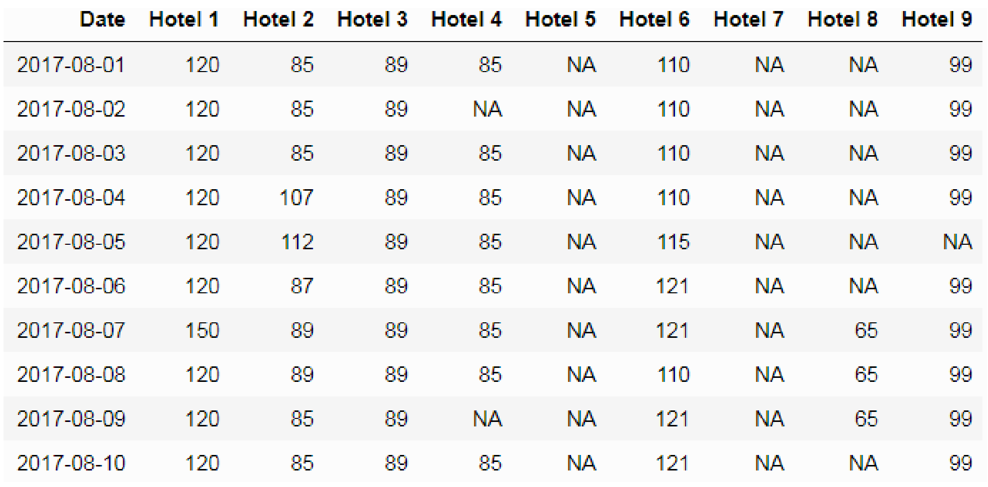
8.1 Preparing the data

The time series have to be entered into the `TSdist` and `TSclust` packages in numerical form (either as a matrix, a list, `data.frame`, `ts` object,...). We therefore modified our input `data.frame` to perform the task. For this we used the function `dcast` in the `reshape2` package [34], through which 3 variables from the input `data.frame` were selected: `NAME`, `MEDIAN` and `DATE`. After selecting the three variables, a new `data.frame` was created where the columns represent the `DATE` and the names of the hotels and guesthouses.

The following code was used:

```
dcast(df[,c("DATE","NAME","MEDIAN")],DATE~NAME,value.var="MEDIAN")
```

After producing the table, the next step was to solve the missing values. As we see in Figure 8.1, when creating the table different missing `NA` values have been obtained for different reasons (due to being unable to find the price for a certain day, because it was closed, etc.) This poses a problem because the `TSclust` and `TSdist` functions require complete time series, i.e. without missing values.



Date	Hotel 1	Hotel 2	Hotel 3	Hotel 4	Hotel 5	Hotel 6	Hotel 7	Hotel 8	Hotel 9
2017-08-01	120	85	89	85	NA	110	NA	NA	99
2017-08-02	120	85	89	NA	NA	110	NA	NA	99
2017-08-03	120	85	89	85	NA	110	NA	NA	99
2017-08-04	120	107	89	85	NA	110	NA	NA	99
2017-08-05	120	112	89	85	NA	115	NA	NA	NA
2017-08-06	120	87	89	85	NA	121	NA	NA	99
2017-08-07	150	89	89	85	NA	121	NA	65	99
2017-08-08	120	89	89	85	NA	110	NA	65	99
2017-08-09	120	85	89	NA	NA	121	NA	65	99
2017-08-10	120	85	89	85	NA	121	NA	NA	99

Figure 8.1: Example of `data.frame` obtained using the function `dcast`.

The following measures were taken with regard to the missing values:

- The hotels and guesthouses with a value of less than 150 (less than 5 months' worth of data) have been eliminated from the analysis.

- On the days they were closed, they were assigned the same price as on the last day they were open, via the function `na.locf` (in the `imputeTS` package).

- The rest of the missing values were imputed via the function `na.seadec` which we saw in part 7.4, using the `kalman` algorithm.

Once the missing data have been imputed, two different clustering were created with the packages `TSclust` and `TSdist` (with the absolute and normalised prices, respectively), as well as another type of clustering to analyse the price variability.

8.2 Clustering with absolute prices

With this first type of clustering, the hotel establishments were grouped by absolute price. Euclidean distance was used to group the series because, among other reasons, it is simple to understand and because it gave the desired results. The *Partitioning Around Medoids*[1] (`pam`) method was to classify the series from the `cluster` package [23], based on the k-means method from section 2.1. The main difference with respect to the k-means method is that the centroid in the cluster has to be one of its elements (medoid).

We used the *silhouette* method to select the number of clusters (using the `pam` algorithm), based on the concept from section 2.1. The silhouette method is used to

[1]In section 2 of [18] the details of the algorithm are explained.

measure the solidity of the clusters, and the silhouette value of element i is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{8.1}$$

where

- $a(i)$: mean distance of element i from the other elements (the same distance used for grouping).
- $b(i)$: mean distance of element i from all elements in the nearest cluster (that do not belong to the same set).

The silhouette method thus calculates the value $s(i)$ for each element and the average of all the values is then obtained. A graph is created with the results, as shown in Figure 8.2.

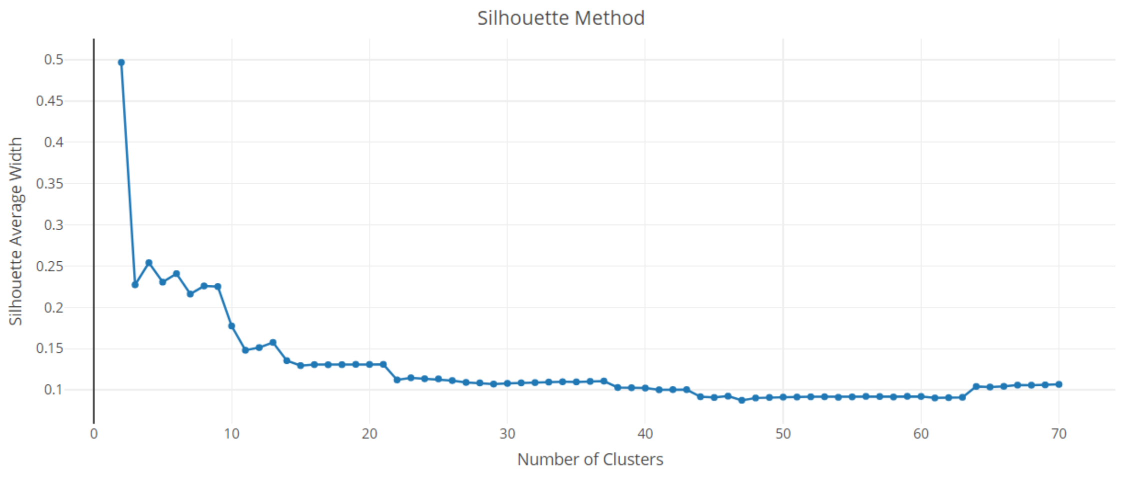


Figure 8.2: Silhouette graph using absolute price clustering.

Once the graph has been drawn, one of the ways of selecting the number of clusters is a point that starts to dampen the decline. In this case, it is the third point (note that the graph starts from 2 because the `pam` function does not permit one sole group to be formed). It is worth remembering that the silhouette method is a criterion and it does not require a specific number of groups to be taken. In our case, the time series are first separated into three groups (the medoids of these groups appear in Figure 8.3). However, as little information was obtained, it was tested with another two groupings, which were those with 7 and 10 clusters (since they were downward points on the graph).

Once the last two clustering were analysed, we decided to merge our time series into a total of seven groups, since relevant information was obtained for each group and because division into ten groups meant that some of them proved very similar (practically indivisible). The medoids of the results obtained can be seen in Figure 8.4. We immediately notice that one group is very different from the other (the one

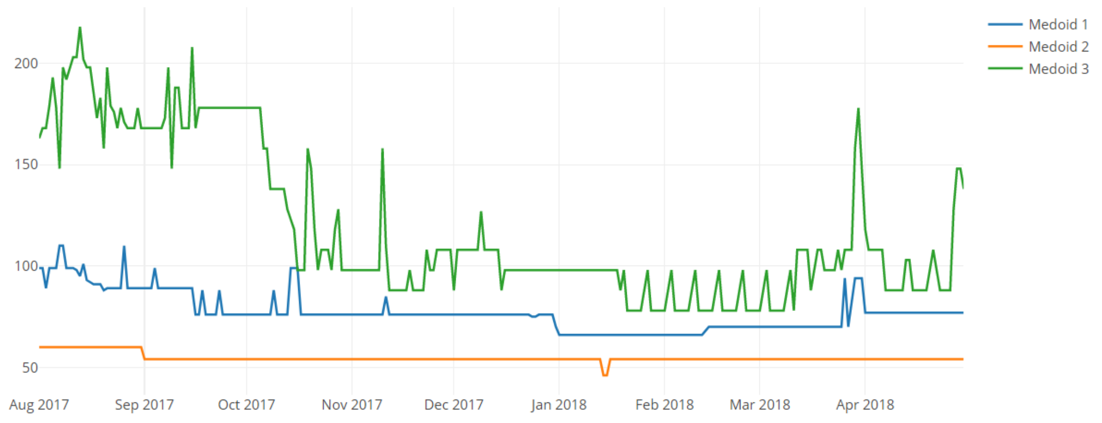


Figure 8.3: Graph of medoids obtained from absolute price clustering (when clustered into three groups).

created using hotels with very high prices). This group comprises very few hotels and, as we might expect, they are luxury hotels.

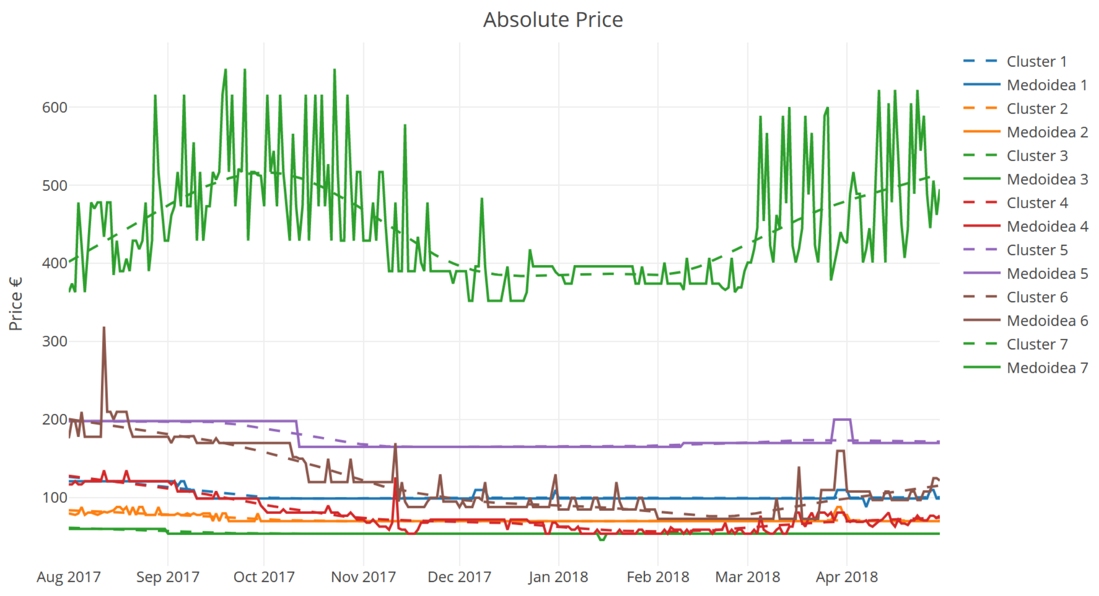


Figure 8.4: Graph showing the medoids of the clusters obtained based on absolute prices (after division into seven groups). The continuous lines are medoids, whereas the discontinuous lines indicate the trend.

Leaving this particular cluster, we will analyse a further three in more detail.

Cluster 5

In this group we find the more expensive hotels and guesthouses (aside from the luxury hotels in cluster 7). Most are 4 and 5 star hotels, as was to be expected (more than 25% of the 4 and 5 star hotels are in this group). It is the second smallest group, with a total of 22 establishments. The price ranges between 300-180€. Figure 8.5 shows several characteristics of this group.

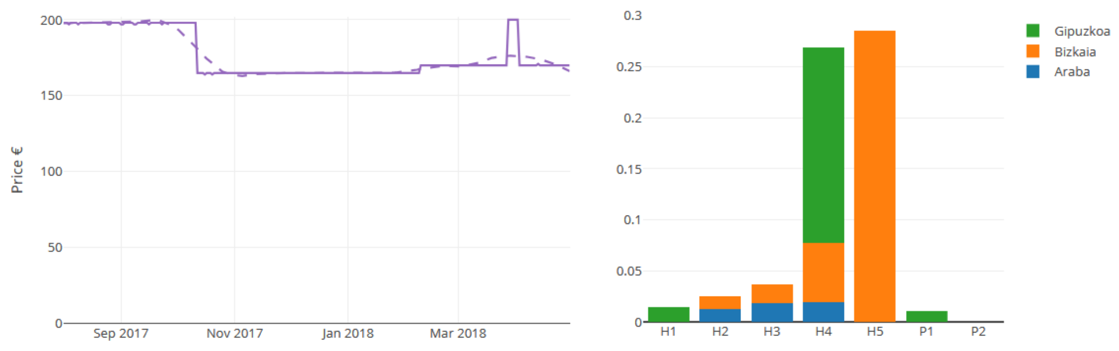


Figure 8.5: The graph on the left shows the medoid of cluster 5 and its trend, which varies between 200-170€. We can see the percentages of the hotel establishment categories.

Cluster 4

The absolute price of these hotels is around 130-60€ . The prices generally vary greatly since the time of year and the season start to have a major impact. A total of 90 hotels and guesthouses are in this group (they offer more than 20% of the hotel beds in the Basque Country[2]). Most of them are in San Sebastián and its region. They are generally guesthouses, and 40% of the 2 star guesthouses and more than 25% of the 1 star guesthouses fall within this group, as we can see in Figure 8.6. From this graph we can extract another interesting piece of information, which is that there are no establishments with these characteristics in inland Gipuzkoa or Bizkaia.

Cluster 3

In absolute prices, these hotels and guesthouses cost around 60-50€. They are cheaper and their prices barely change according to the time of year. As they are cheaper, it is understandable that most would have fewer than three stars and that they generally offer few rooms. They are therefore small establishments: there are a total of 104 hotels and guesthouses in this group, offering less than 15% of the total rooms. There are barely any establishments with these characteristics in Rioja Alavesa and San Sebastián, while most of the hotels in Álava (aside from Vitoria-Gasteiz and Rioja Alavesa) – more than 60%– are in this group, as we can see in Figure 8.7.

[2]The percentage has been calculated from the total number of rooms offered by the 443 establishments offered on the web.

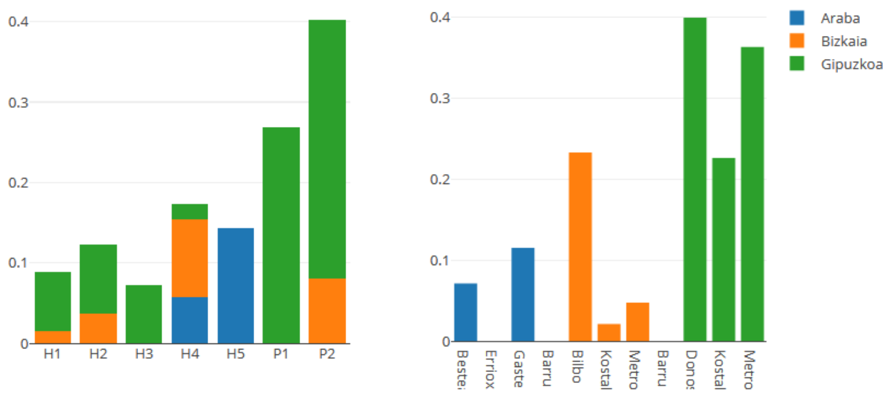


Figure 8.6: The graph on the left shows the category of the hotel establishments in cluster 4, whereas the graph on the right shows the strata.

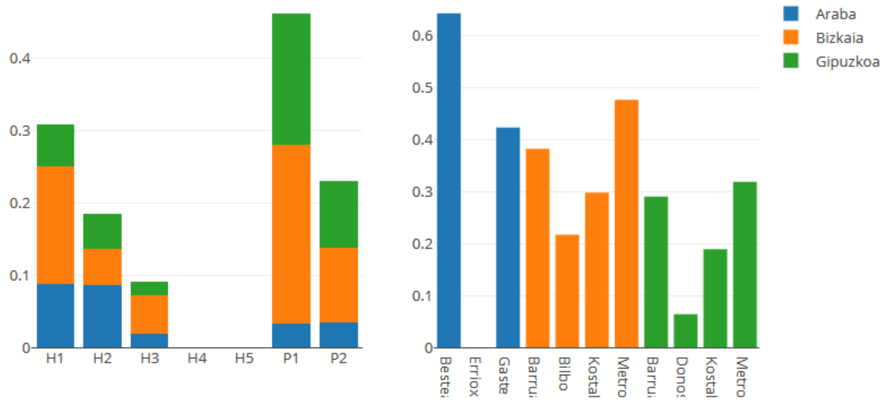


Figure 8.7: The graph on the left shows the category of the hotel establishments in cluster 3, whereas the graph on the right shows the strata.

8.3 Clustering with normalised prices

With this second type of clustering, the hotel establishments were grouped by normalised price. The *Min-Max* method was used for this; in other words, the scale of all hotel prices was changed to values between 0 and 100. The following normalisation formula was used for every establishment:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \cdot 100 \tag{8.2}$$

- x: Price vector of the establishment
- x_i: Price of the establishment on day i
- z_i: Normalised price of the establishment on day i, which will be in the range $[0, 100]$

The scale is changed in order to compare all hotels and guesthouses in the same way, so that the trends can then be analysed and they can be classified based on that trend. Figure 8.8 shows a relatively clear example. Both guesthouses have quite different prices, but the trend is very similar. In this case, we would expect these two guesthouses to be classified in the same group.

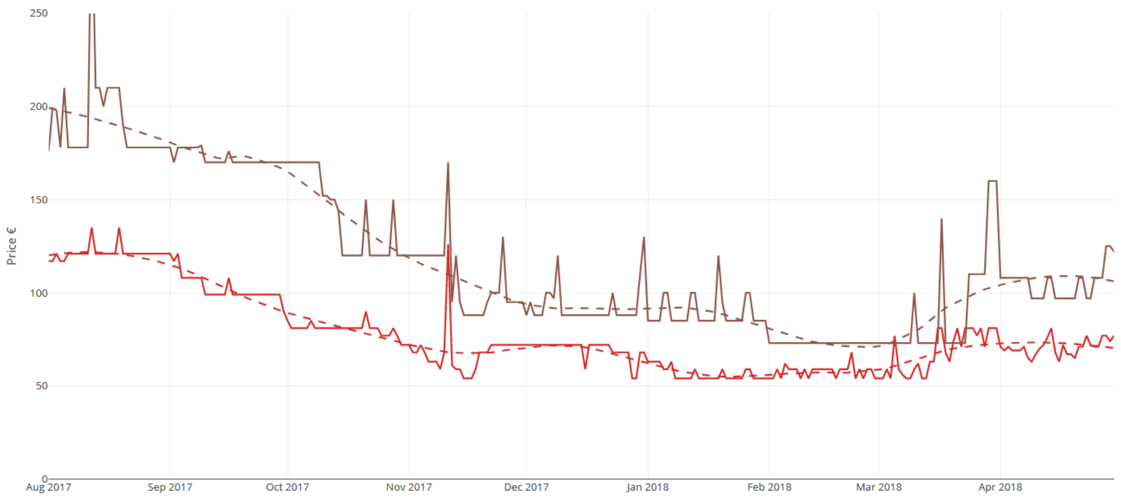


Figure 8.8: This graph shows the different oscillating price ranges of the guesthouses. However, it is clear that they both follow quite a similar trend.

When working with the data, we might have problems when applying the formula (8.2) since the cases $\max(x) = \min(x)$are indeterminate. In this case, we have to deal with the establishments with constant prices, which have been assigned the constant value of 100 (they were tested with a total of three different values, 0, 50 and 100, and the most satisfactory results were obtained with 100). Furthermore, all the establishments with a maximum price variation of less than 5€ have been assigned a value of 100. A variation of 5€ has been regarded as the *minimum*, and with *Min-Max* normalisation these same establishments have much higher oscillations.

After normalising the prices, we continued with the same clustering method as before, using Euclidean distance. The resulting silhouette graph is shown in Figure 8.9. In this case, we can see the cut-off point clearly and simply, for which we have created a total of four groups (remember that the first represents 2 clusters).

The results obtained with these four clusters can be seen in Figure 8.10 and they can be distinguished relatively easily. Group 4, as defined before, comprises hotels and guesthouses with constant or practically constant prices. They are located inland and have low ratings. On the other hand, prices in other groups vary according to season. Those with the most variation are included in the first cluster, while those with the least are grouped in the third cluster. In this group, there are establishments whose price trend barely changes but, compared with those in group 4, their prices do change on special days of the year and/or at weekends.

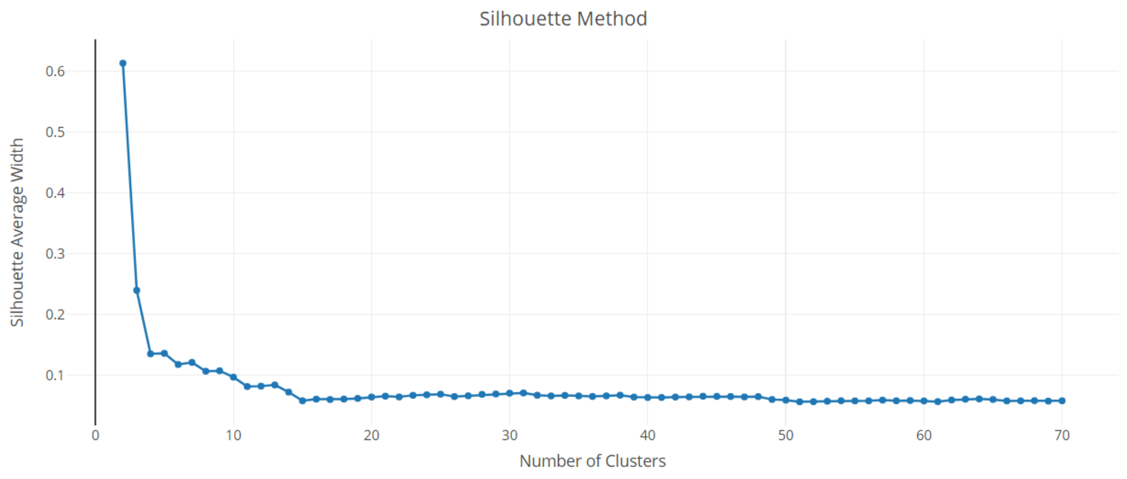


Figure 8.9: This graph displays the results obtained from the silhouette method. We can see that the silhouette value reduces considerably with 3 and 4 clusters, and the improvement is insignificant from then on.

Cluster 1

This cluster contains hotels and guesthouses that show the most marked change in normalised price trends. If we look at Figure 8.11, we will see that the 4 star hotels and guesthouses in this group carry significant weight. Practically 70% of hotels in San Sebastián belong to this group. Furthermore, almost 50% of the hotel establishments on the coast of Gipuzkoa and in the metropolitan areas of Donostia and Bilbao are found in this group. There are a total of 191 hotels and guesthouses with these characteristics and they make up around 45% of the total available rooms.

Cluster 2

There is a high incidence of 3, 4 and especially 5 star hotels in this group. In terms of strata, 45% of establishments on the coast of Gipuzkoa are included in this group and, as we can see, most are located outside urban centres. This group comprises a total of 130 establishments, which account for 35% of rooms.

Custer 3

Most of the hotels in this group have 1, 2 and 5 stars. The prices of these establishments show small fluctuations, i.e. the season does not have a major influence on the prices offered. However, the prices also change depending on the day, during the week in some cases and on special days in others (public holidays, special celebrations). Regarding the strata, Álava (except Rioja Alavesa) and inland Bizkaia offer this type of hotel/guesthouse in particular.

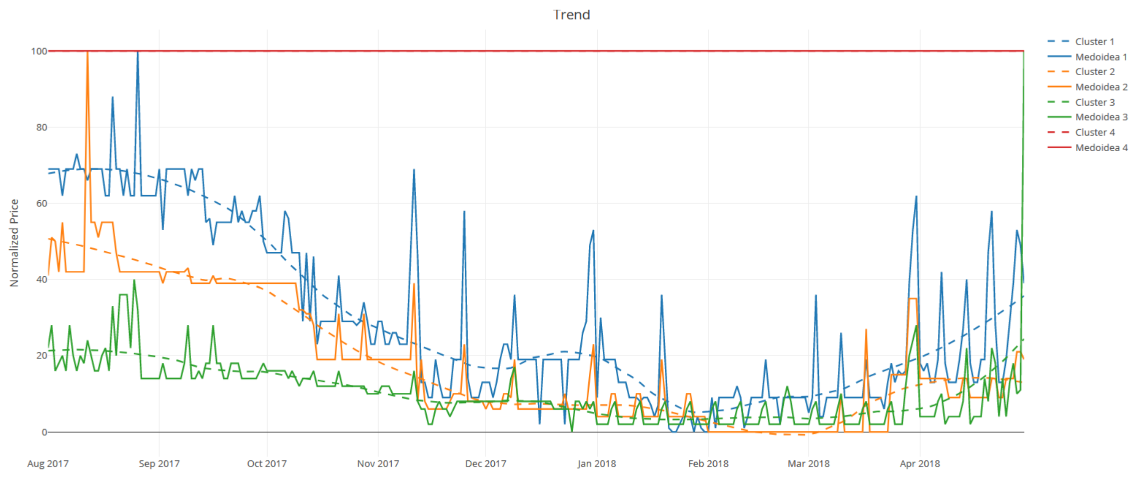


Figure 8.10: Clustering obtained with normalised prices. The continuous lines represent the medoids, whereas the discontinuous lines are the medoid trend.

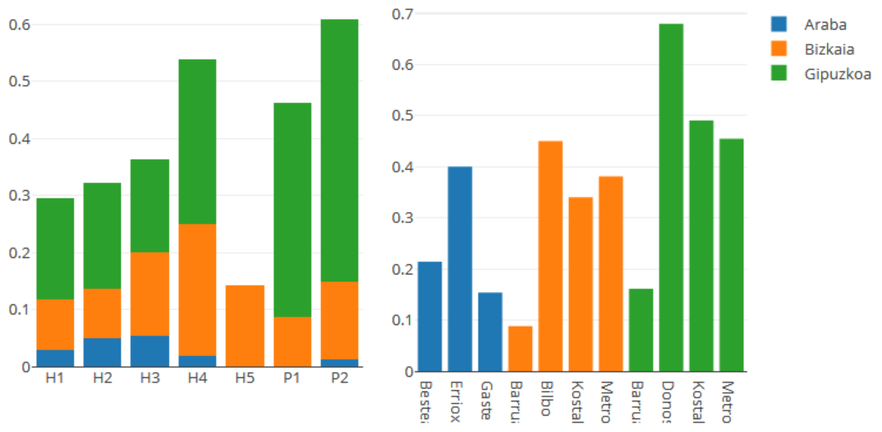


Figure 8.11: Classification of the hotel establishments in cluster 1, by category and stratum.

8.4 Volatility clustering

This last type of clustering is based on volatility in order to measure the price change from one day to the next. The main goal of this grouping is to classify hotels and guesthouses that change their prices considerably, little or not at all in the short term. The study is based on a measurement used in economics. **Close-to-Close Volatility** or **Close/Close Volatility** [1].

The main idea behind this measure is to analyse the difference between consecutive days using the logarithm and then work out the standard deviation of those values. For example, let us suppose that we have the following price vector:

$$90, 90, 90, 90, 100, 105, 80, 90, 90, 90, 90, 100, 105, 80$$

Firstly, we apply the logarithm to these values (so that an establishment charging

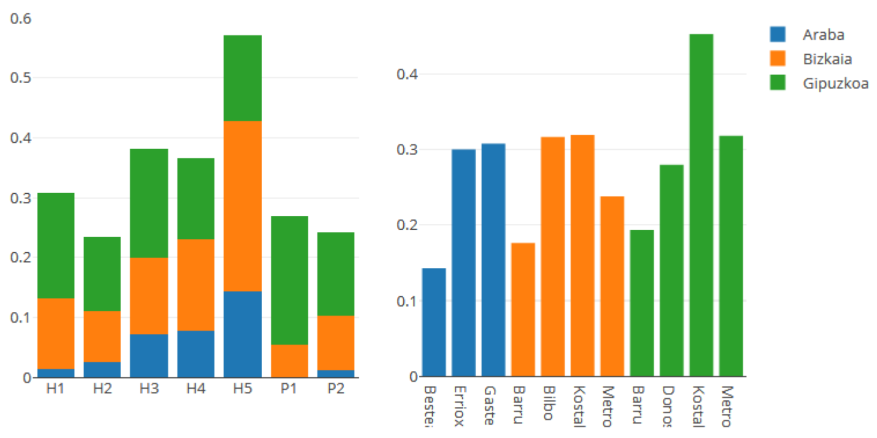


Figure 8.12: Classification of the hotel establishments in cluster 2, by category and stratum.

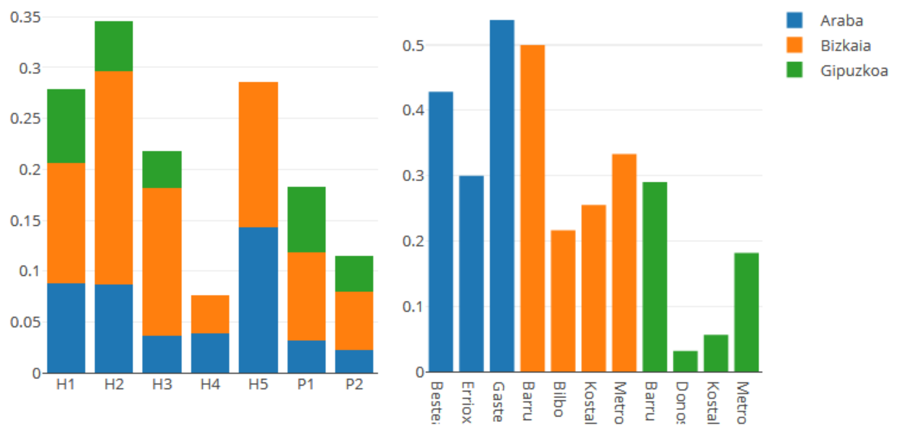


Figure 8.13: Classification of the hotel establishments in cluster 3, by category and stratum.

80€ and another charging 200€ will shift in similar scales):

$$4.5, 4.5, 4.5, 4.5, 4.605, 4.654, 4.382, 4.5, 4.5, 4.5, 4.5, 4.605, 4.654, 4.382$$

the sequential difference is then calculated (we should take into account that the length of the vector in a unit will be reduced):

$$0, 0, 0, 0.105, 0.049, -0.272, 0.118, 0, 0, 0, 0.105, 0.049, -0.272$$

and, finally, the standard deviation is calculated.

The same idea is used for this task except that instead of calculating the standard deviation at the end, the mean of the absolute values is calculated, since it is simpler and is sufficient for our analysis. We calculated the volatility value used for our analysis as follows:

```
mean(abs(diff(log(x)))
```

where x represents a price vector. Once this process has been applied to all hotel establishments, the values were displayed to see whether or not we notice anything at first glance. Nonetheless, since we cannot draw any well-founded conclusions regarding clustering from the visualisation of the boxplot in Figure 8.14 , we established the groups using quantiles. We experimented with 3, 4 and 5 groups, and finally decided to select 4 (quartiles).

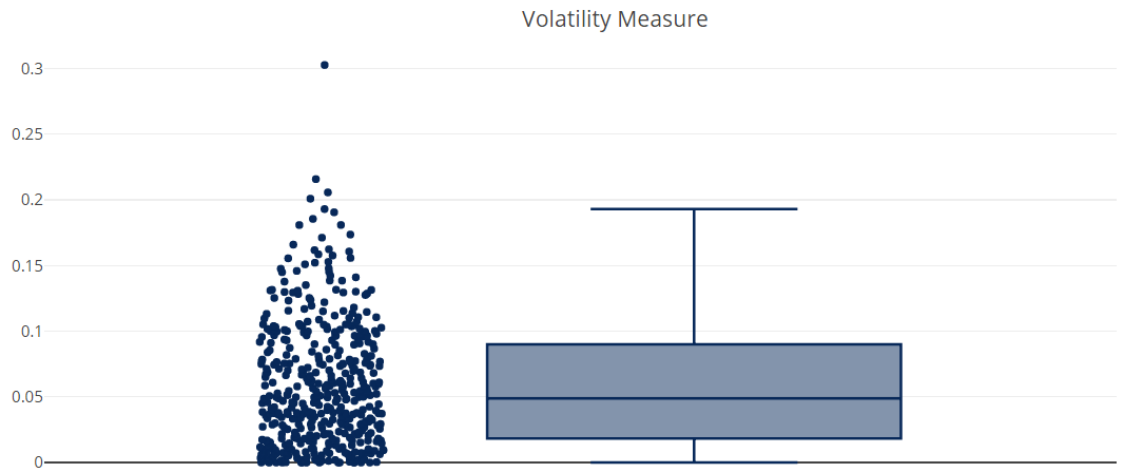


Figure 8.14: Boxplot of the values obtained by applying the close-to-close algorithm to the hotels and guesthouses.

In this case, the medoids are not specified for each cluster, given the construction method. However, with the aim of maintaining the style of the previous graphs, a representative establishment was selected from each of the groups (we tried different options and selected the most visually appropriate ones). The results are shown in Figure 8.15 (ordered from least to most volatile).

Three groups will then be analysed in further detail, as we have done in previous cases.

Cluster 1

Here we find the establishments with the least volatility, i.e. those that rarely or never change their prices. As we can see in Figure 8.16 most are hotels and pensions with three stars or fewer (more than 40% of the one-star establishments are in this group) and are usually located inland. More than 75% of the establishments in inland Gipuzkoa and Álava (except Rioja Alavesa and Vitoria-Gasteiz) have very low price volatility. Nevertheless, this group offers only 15% of rooms, suggesting that they are small establishments.

Cluster 3

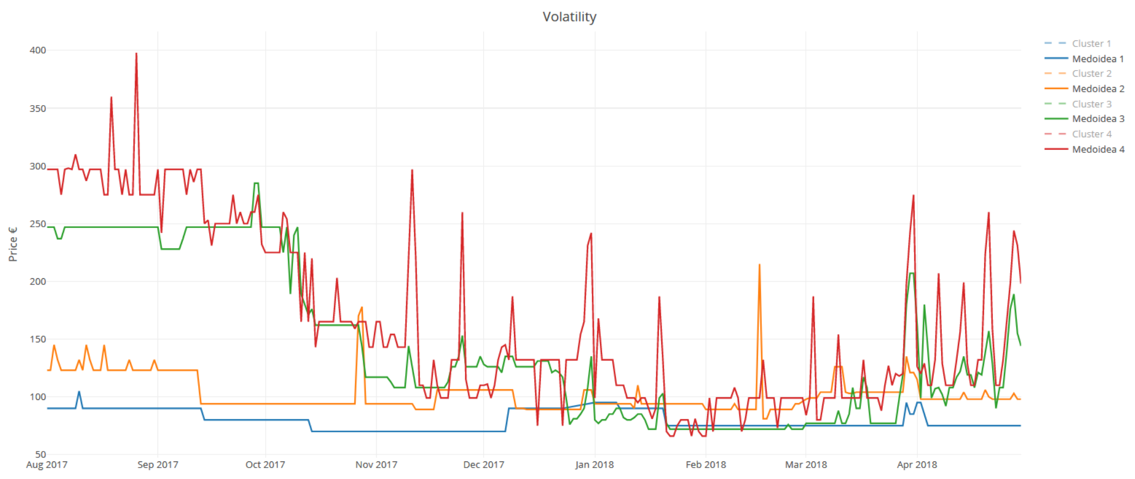


Figure 8.15: Clustering according to volatility. The clusters are arranged from least to most volatile.

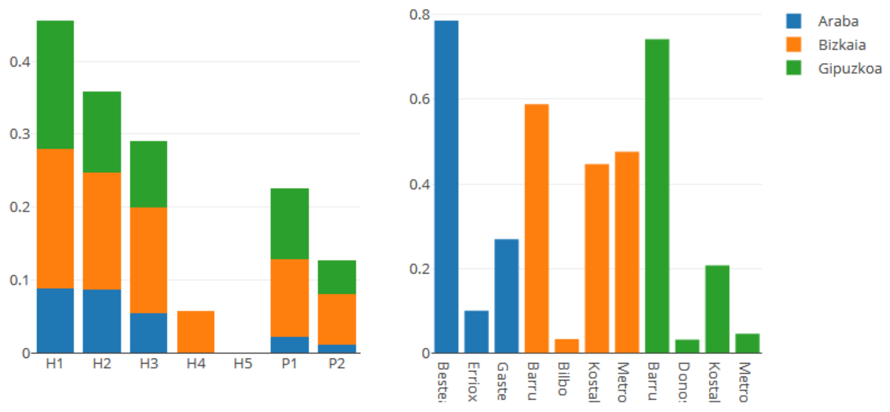


Figure 8.16: Classification of the hotel establishments in cluster 1, by category and stratum.

Very few inland hotels and guesthouses appear in this third group; more than 40% of them are located in Bilbao and Rioja Alavesa (as can be seen in Figure 8.17). They are high-ranking hotels with three, four and five stars. All of the five star hotels in Bilbao are in this group. These establishments have high price volatility, but their prices do not vary the most.

Cluster 4

These are establishments with the most price volatility. It is clear from Figure 8.18 that most are guesthouses located in San Sebastián (more than 60%). On the other hand, they offer only 16% of rooms, meaning that they are small.

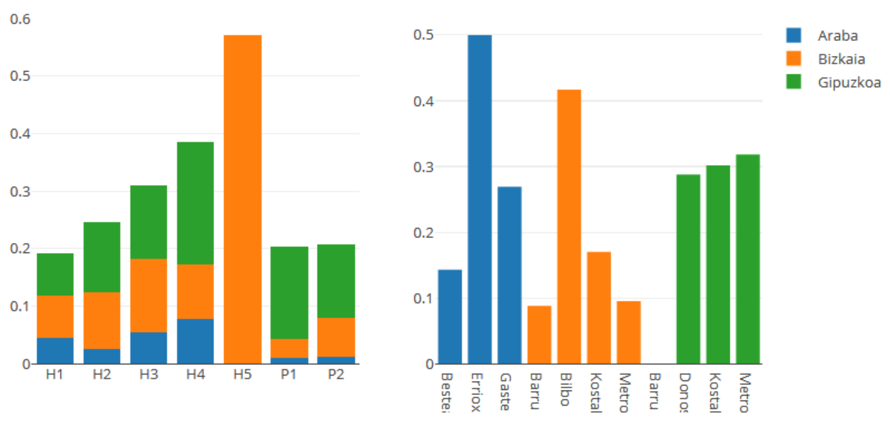


Figure 8.17: Classification of the hotel establishments in cluster 3, by category and stratum.

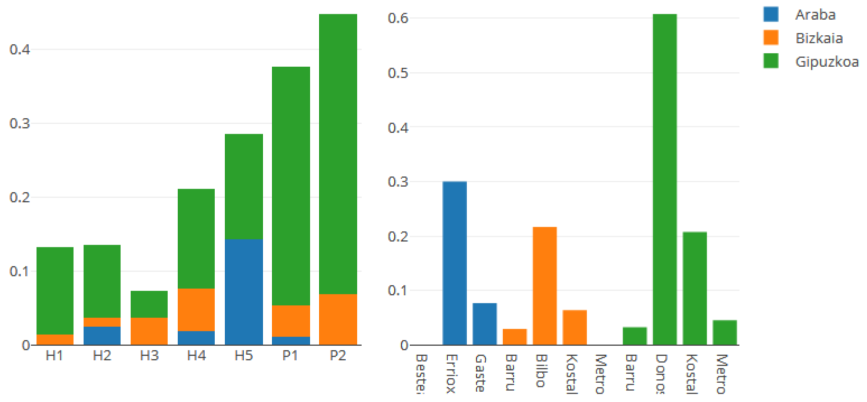


Figure 8.18: Classification of the hotel establishments in cluster 4, by category and stratum.

Bibliography

[1] Bennett, C., and Gil, M. A. Measuring historical volatility.

[2] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] Canavos, G. *Probabilidad y Estadística.* McGraw Hill, 1994.

[4] Cappé, O., Moulines, E., and Ryden, T. *Inference in Hidden Markov Models (Springer Series in Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[5] Casella, G., and Berger, R. *Statistical Inference.* Duxbury Resource Center, June 2001.

[6] Chang, W., Cheng, J., Allaire, J., Xie, Y., and McPherson, J. *shiny: Web Application Framework for R*, 2017. R package version 1.0.5.

[7] Chen, E. Winning the Netflix Prize: A Summary. `http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/`.

[8] Cheng, J., Karambelkar, B., and Xie, Y. *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*, 2018. R package version 2.0.0.

[9] de Lacalle, J. L. *tsoutliers: Detection of Outliers in Time Series*, 2017. R package version 0.6-6.

[10] Devore, J. L. *Probability and Statistics for Engineering and the Sciences*, 8th ed. Brooks/Cole, January 2011. ISBN-13: 978-0-538-73352-6.

[11] Garey, M. R., and Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[12] Giorgino, T. Computing and visualizing dynamic time warping alignments in R: The dtw package. *Journal of Statistical Software 31*, 7 (2009), 1–24.

[13] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[14] Grubbs, F. E. Sample criteria for testing outlying observations. *Ann. Math. Statist. 21*, 1 (03 1950), 27–58.

[15] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[16] Hyndman, R. J. *forecast: Forecasting functions for time series and linear models*, 2017. R package version 8.2.

[17] James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated, 2014.

[18] Kaufman, L., and Rousseeuw, P. *Finding Groups in Data: an introduction to cluster analysis.* Wiley, 1990.

[19] Komsta, L. *outliers: Tests for outliers*, 2011. R package version 0.14.

[20] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer 42*, 8 (Aug 2009), 30–37.

[21] Kriesel, D. *A Brief Introduction to Neural Networks.* 2007.

[22] Levin, D. A., Peres, Y., and Wilmer, E. L. *Markov chains and mixing times.* American Mathematical Society, 2006.

[23] Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. *cluster: Cluster Analysis Basics and Extensions*, 2018. R package version 2.0.7-1 — For new features, see the 'Changelog' file (in the package source).

[24] Montero, P., and Vilar, J. A. TSclust: An R package for time series clustering. *Journal of Statistical Software 62*, 1 (2014), 1–43.

[25] Mori, U., Mendiburu, A., and Lozano, J. A. Distance measures for time series in r: The TSdist package. *R journal 8*, 2 (2016), 451–459.

[26] Moritz, S. *imputeTS: Time Series Missing Value Imputation*, 2017. R package version 2.5.

[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[28] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2017.

[29] Rabiner, L. R. Readings in speech recognition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, ch. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296.

[30] Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. *Recommender systems handbook.* Springer, New York; London, 2011.

[31] Shumway, R., and Stoffer, D. *Time Series Analysis and Its Applications: With R Examples.* Springer Texts in Statistics. Springer International Publishing, 2017.

[32] Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M., and Despouy, P. *plotly: Create Interactive Web Graphics via 'plotly.js'*, 2017. R package version 4.7.1.

[33] Strang, G. *Linear algebra and its applications.* Thomson, Brooks/Cole, Belmont, CA, 2006.

[34] Wickham, H. Reshaping data with the reshape package. *Journal of Statistical Software 21*, 12 (2007), 1–20.

Organismo Autónomo del

EUSKO JAURLARITZA
GOBIERNO VASCO

Eustat
EUSKAL ESTATISTIKA ERAKUNDEA
INSTITUTO VASCO DE ESTADÍSTICA
www.eustat.es

www.eustat.es